contrôle et implantation des systèmes répartis de fusion d'informations

Olivier Passalacqua

THESE

présentée et soutenue publiquement le 02 décembre 2009 à Annecy-le-Vieux

en vue du Doctorat de

l'UNIVERSITE DE SAVOIE

mention STIC Informatique

par

M. Olivier PASSALACQUA

Contrôle et implantation des systèmes répartis de fusion d'informations.

Thèse préparée au LISTIC et encadrée par : Pr Patrice MOREAUX
Dr Éric BENOIT
Dr Marc-Philippe HUGET

COMPOSITION DU JURY

Président :	Pr David HILL, LIMOS, Univ. Blaise Pascal, Clermont-Ferrand
Rapporteurs :	Pr Jean-François PRADAT-PEYRE, LIP6, Univ. Paris Ouest la Défense, Paris
	Pr Jean-Marc THIRIET, GIPSA-Lab, Univ. Joseph Fourier, Grenoble
Examinateurs :	Dr Stéphane FRENOT, CITI INSA, Univ. Lyon, Lyon
	Pr Patrice MOREAUX, LISTIC, Univ. de Savoie, Annecy
	Dr Éric BENOIT, LISTIC, Univ. de Savoie, Annecy
	Dr Marc-Philippe HUGET, LISTIC, Univ. de Savoie, Annecy

THESE

présentée et soutenue publiquement le 02 décembre 2009 à Annecy-le-Vieux

en vue du Doctorat de

l'UNIVERSITE DE SAVOIE

mention STIC Informatique

par

M. Olivier PASSALACQUA

Contrôle et implantation des systèmes répartis de fusion d'informations.

Thèse préparée au LISTIC et encadrée par : Pr Patrice MOREAUX
Dr Éric BENOIT
Dr Marc-Philippe HUGET

COMPOSITION DU JURY

Président :	Pr David HILL, LIMOS, Univ. Blaise Pascal, Clermont-Ferrand
Rapporteurs :	Pr Jean-François PRADAT-PEYRE, LIP6, Univ. Paris Ouest la Défense, Paris
	Pr Jean-Marc THIRIET, GIPSA-Lab, Univ. Joseph Fourier, Grenoble
Examinateurs :	Dr Stéphane FRENOT, CITI INSA, Univ. Lyon, Lyon
	Pr Patrice MOREAUX, LISTIC, Univ. de Savoie, Annecy
	Dr Éric BENOIT, LISTIC, Univ. de Savoie, Annecy
	Dr Marc-Philippe HUGET, LISTIC, Univ. de Savoie, Annecy

à Gino Passalacqua, parti lors de mon séjour de recherche au Japon.

Remerciements

Cette thèse a été menée au sein du Laboratoire d'Informatique, Systèmes et Traitement de l'Information et de la Connaissance (LISTIC) de l'Université de Savoie. Mes premiers remerciements vont donc naturellement à ces deux institutions qui m'ont accueilli dans leurs locaux. Je remercie également messieurs Jean-Marc Thiriet et Jean-François Pradat-Peyre pour leurs suggestions, proposées dans leurs pré-rapports de soutenance et lors de nos discussions, me permettant ainsi d'améliorer ce mémoire.

La soutenance d'une thèse est un événement unique dans la vie d'un étudiant, et elle marque la fin d'un chapitre, bien plus symbolique que ceux qui jalonnent ce mémoire. En effet, la rédaction de ce mémoire ne représente que six des quarante mois durant lesquels, messieurs Patrice Moreaux, Éric Benoit et Marc-Philippe Huget, m'ont accompagné jusqu'à l'achèvement de mon Doctorat. J'ai donc une sincère gratitude pour ces trois chercheurs, aux personnalités si différentes, avec qui j'espère encore partager des moments, tantôt rigoureux et constructifs, lors des échanges scientifiques, et tantôt légers, dès qu'une parenthèse s'ouvre sur un de nos centres d'intérêt.

La préparation de ma thèse et les travaux de recherche m'ont mené jusqu'au Japon où j'ai eu la chance de découvrir une culture aux antipodes de la nôtre, et un peuple bienveillant une fois la barrière de la langue franchie. Je tiens ici à remercier le Ministère des Affaires étrangères et européennes, par l'intermédiaire de monsieur Yves Mansuy, et la région Rhône-Alpes, pour leurs aides financières dans ce projet. J'ai une pensée particulière pour monsieur et madame Tokai, mes parents d'accueil à Takamatsu, pour monsieur Hydeyuki Sawada ainsi que les membres de son laboratoire, en particulier messieurs Kitani et Wan, et enfin pour madame Ai Nishioka de l'Université de Kagawa.

Les journées de travail au laboratoire, en l'occurrence en B206, ont été le théâtre de certaines élucubrations musicales, et je laisse dans ces mots un dernier hommage à la musique *dance*. Je tiens à saluer le courage de messieurs Grégory Païs, Nabil Fakhfakh, Florent Martin et Amory Bisserier pour avoir supporté mes extravagances, qu'ils ont parfois partagées. La vie est ainsi faite que, au détour d'une coïncidence, des chemins se croisent et donnent naissance à une rencontre, comme celle d'Amory. Je suis très heureux d'avoir croisé la route de ce futur docteur d'ores et déjà chercheur qui, bien plus que « le collègue qu'il vous faut », demeurera surtout un ami avec qui j'espère maintenir les liens que nous avons tissés.

Mes derniers remerciements vont à ceux pour qui ces trois dernières années ont été comme une longue semaine de travail marquée par mon absence. Je pense bien sûr à mes parents et à ma sœur. Mais je pense surtout à Marie qui a eu le courage de me suivre de Clermont-Ferrand à Annecy, dans un deux pièces hors de prix, et même jusqu'au Japon où nous avons vécu des moments riches en souvenirs. Cette femme merveilleuse, devenue entre temps mon épouse, est la personne avec qui je commence un nouveau chapitre de ma vie, cette fois intitulé "contrôle des biberons et mise en œuvre avec un nouveau né".

Mots-clé : Fusion d'informations, contrôle des systèmes répartis, analyse automatique de performances, répartition bio-inspirée par système multi-agents

Résumé : La fusion d'informations est une discipline bien plus ancienne que l'informatique moderne, et on en retrouve des applications dès lors qu'il est nécessaire de regrouper des informations potentiellement imprécises ou incertaines. Initialement réalisée de façon mentale, la fusion d'informations nécessite à présent une mise en œvre sur un support d'exécution informatisé, dénommé système de fusion, dont les évolutions technologiques ont guidée la répartition des ressources selon des contraintes géographiques, physiques et de sécurité.

Cette thèse présente une étude du contrôle des systèmes répartis de fusion d'informations. Elle propose notamment un système de contrôle capable d'adapter l'utilisation des ressources disponibles au processus de fusion (PF) mis en œvre, assez générique pour permettre l'expression des modèles de fusion les plus courants - en l'occurence des modèles de fusion basés sur des graphes de flot de données. Cinq tâches de contrôle sont dédiées à l'accomplissement d'un sous-objectif. La première consiste en la recherche d'une répartition des éléments du PF, basée sur le parcours d'un espace des répartitions possibles construit selon les ressources disponibles. Vient ensuite le déploiement effectif d'un PF sur un ensemble variable de ressources réparties. Pendant l'exécution du PF, la tâche d'analyse des performances du système permet d'adapter la répartition pour le rendre plus performant, grâce à un modèle de la répartition traduit en réseau de Petri stochastique généralisé. En parallèle, une autre tâche de contrôle surveille l'ensemble du système réparti pour détecter l'ajout de ressources, mais aussi les occurrences d'erreurs pouvant conduire à des défaillances. Enfin, la décision finale d'une modification de la répartition, soit pour en améliorer les performances, soit pour corriger la répartition courante et maintenir l'exécution du processus, découle d'une phase de négociation au sein d'un système multi-agents. L'ensemble du système de contrôle tire profit des ressources réparties par un mécanisme bio-inspiré, dans lequel la place de chaque agent est guidée par sa puissance de calcul et par ses interactions avec les autres agents du système. A titre expérimental, un système de fusion d'informations nommé eZFusion, implémentant les tâches de déploiement et d'analyse des performances, a été réalisé sur un réseau de plate-formes OSGI.

Abstract : Information fusion is almost as old as mankind and it is today a research domain of the computer science. Its use is required since potential uncertain or imprecise data should be combined. The fusion operations have to be performed on physical resources, named Information Fusion Systems (IFS), since fusion processes are now too complex. IFS are now distributed over communicating devices due to geographical or security issues. The purpose of this Ph. D. thesis deals with the control of distributed IFS. Especially it presents a control system able to adapt the use of available resources to the fusion process requirements. Thus, the description of the fusion process, built upon a data flow graph, is generic enough to cover the scope of the common information fusion models. Five control tasks compose the control system in order to reach five dedicated goals. The first task is in charge of matching each part of the process to the compatible resources. Then, the configuration built from the previous assignment, is effectively deployed on the distributed devices. During the execution of the fusion process, two other tasks operate : the first one analyses the system performance in order to improve it, and the second one monitors the execution and detects faults before failures occur. The final decision about updating the configuration comes from a negotiation between agents. The whole control system makes profit of the distributed resources thanks to a bio-inspired mechanism in which each agent infers its location in a control hierarchy from its CPU power and from its interactions with its neighbours. From a practical viewpoint, two tasks of the control system, respectively the one in charge of the deployment and the one in charge of the performance analysis, are implemented in eZFusion that is a distributed IFS developed over a network of OSGi frameworks.

Table des matières

1 Introduction générale **21**

2 Problématique et objectifs de la thèse **25**
2.1 Systèmes de fusion d'informations . . . 25
2.1.1 Exemple de fusion d'informations . . . 25
2.1.2 Origine des systèmes de fusion d'informations . . . 26
2.1.3 Évolutions des systèmes de fusion d'informations . . . 26
2.1.4 Nouvelles problématiques des systèmes de fusion d'informations . . . 27
2.2 Définitions . . . 28
2.2.1 Définition d'un système de fusion d'informations . . . 29
2.2.2 Systèmes répartis de fusion d'informations . . . 29
2.3 Gestion de la volatilité des ressources . . . 32
2.4 Adaptation de la puissance de calcul allouée au traitement . . . 35
2.5 Passage à l'échelle du système . . . 35
2.6 Réutilisation et généricité du système . . . 36
2.7 Synthèse . . . 37

I État de l'art **39**

3 Étude des systèmes répartis de fusion d'informations **41**
3.1 Le matériel de mise en œuvre . . . 41
3.2 Les outils de mise en œuvre . . . 42
3.2.1 Les systèmes d'exploitation dédiés . . . 43
3.2.2 Les cadres d'exécution dédiés . . . 43
3.2.3 Les plates-formes de fusion d'informations . . . 44
3.2.4 Les plates-formes de services de fusion d'informations . . . 44
3.3 Synthèse . . . 46

4 Étude des modèles de fusion **47**
4.1 Les paradigmes de mise en œuvre . . . 47
4.1.1 Réseaux neuronaux . . . 47
4.1.2 Producteur-consommateur . . . 48
4.1.3 Systèmes multi-agents . . . 48
4.2 Les modèles de fusion . . . 50
4.2.1 Fusion probabiliste . . . 51
4.2.2 Fusion possibiliste . . . 51
4.2.3 Fusion de fonctions de croyances . . . 52
4.3 La théorie de la fusion d'informations . . . 53

4.4 Synthèse . . . 54

II Travaux théoriques **55**

5 Contrôle du système de fusion **57**
5.1 Système de contrôle . . . 58
5.2 Cycle de contrôle . . . 58
5.2.1 Perception . . . 58
5.2.2 Décision . . . 59
5.2.3 Action . . . 59
5.3 Tâches de contrôle . . . 59
5.3.1 Configuration . . . 60
5.3.2 Implémentation . . . 60
5.3.3 Performance . . . 61
5.3.4 Robustesse . . . 61
5.3.5 Reconfiguration . . . 61
5.4 Synthèse . . . 62

6 De la description à l'implémentation **63**
6.1 Modèles de fusion visés . . . 63
6.1.1 Contrôle du processus . . . 64
6.1.2 Qualité de la fusion . . . 64
6.2 Description du processus de fusion . . . 64
6.2.1 Élément de fusion . . . 64
6.2.2 Processus . . . 66
6.3 Compatibilité des modèles de fusion . . . 67
6.4 Modèle de déploiement du processus de fusion . . . 70
6.4.1 Éléments d'une configuration . . . 70
6.4.2 Attributs d'une configuration . . . 70
6.4.3 Modèle sous-jacent . . . 72
6.5 Gestion de la configuration . . . 73
6.6 Implémentation du processus de fusion . . . 75
6.6.1 Implémentation d'une configuration . . . 75
6.6.2 Interactions des éléments . . . 78
6.7 Gestion de l'implémentation du processus . . . 79
6.7.1 Reconfiguration de l'implémentation . . . 80
6.7.2 Mesures du système de fusion . . . 81
6.8 Synthèse . . . 84

7 Analyse de la configuration **87**
7.1 Choix des indices de performance . . . 87
7.2 Le modèle *Generalized Stochastic Petri Net* . . . 88
7.3 Construction du modèle . . . 89
7.3.1 Nœud de fusion . . . 90
7.3.2 Étape locale . . . 90
7.3.3 Étape distante . . . 90
7.3.4 CPU et réseau . . . 91
7.3.5 Système réparti de fusion d'informations . . . 91
7.3.6 Exemples . . . 91

7.4 Modèle fini . . . 93
7.4.1 Description . . . 93
7.4.2 Construction . . . 93
7.5 Paramètres du modèle . . . 93
7.6 Calcul des indices . . . 95
7.7 Gestion de la performance . . . 96
7.7.1 Création d'un modèle . . . 96
7.7.2 Mise à jour des attributs . . . 97
7.8 Simplification de la recherche d'une configuration . . . 97
7.9 Synthèse . . . 98

8 Surveillance et reconfiguration, propositions 99
8.1 Détection d'erreurs et de défaillances . . . 99
8.1.1 Erreurs et défaillances . . . 99
8.1.2 Prévention . . . 101
8.1.3 Gestion de la robustesse . . . 104
8.2 Reconfiguration . . . 104
8.2.1 Correction d'une erreur ou d'une défaillance . . . 105
8.2.2 Amélioration de la configuration . . . 107
8.2.3 Gestion de la reconfiguration . . . 108
8.3 Synthèse . . . 109

9 Répartition du contrôle 111
9.1 Les types de répartition . . . 111
9.2 Système multi-agents . . . 112
9.2.1 Définition . . . 112
9.2.2 Utilité d'un système multi-agents . . . 115
9.3 Contrôle hiérarchique . . . 116
9.4 Répartition par tâche . . . 116
9.4.1 Parallélisme . . . 117
9.4.2 Hiérarchie multi-niveaux . . . 119
9.5 Deux types d'agents . . . 120
9.5.1 Agent mobile . . . 120
9.5.2 Agent immobile . . . 121
9.5.3 Transformations . . . 121
9.6 Mécanisme de différentiation et d'auto-stabilisation . . . 122
9.6.1 Modèle 1 . . . 123
9.6.2 Modèle 2 . . . 123
9.7 Activation et désactivation d'un niveau de contrôle . . . 124
9.7.1 Interaction de contrôle . . . 124
9.7.2 Limites d'activation . . . 125
9.8 Transition inter-niveaux . . . 126
9.8.1 Niveaux activables . . . 126
9.8.2 Mécanismes d'activation . . . 126
9.8.3 Mécanismes de désactivation . . . 126
9.9 Communication de contrôle . . . 128
9.9.1 Au sein d'une tâche de contrôle . . . 128
9.9.2 Entre deux tâches de contrôle . . . 129
9.10 Synthèse . . . 129

10 Analyse du mécanisme de répartition **131**
10.1 Modèle DEVS . . . 131
10.2 Modélisation des agents . . . 133
10.3 Observations . . . 135
10.3.1 Premières hypothèses . . . 137
10.3.2 Cadre expérimental . . . 138
10.4 Synthèse . . . 143

III Mise en œuvre **145**

11 Implémentation actuelle du système **149**
11.1 Présentation générale . . . 149
11.1.1 Compatibilité visée . . . 149
11.1.2 Systèmes d'exploitations compatibles . . . 149
11.1.3 Infrastructure réseau . . . 150
11.1.4 Installation et utilisation . . . 150
11.2 Cadre de déploiement de eZFusion . . . 150
11.3 Conditionnement du déploiement . . . 151
11.3.1 *Bundles* . . . 151
11.3.2 Services . . . 152
11.4 Éléments d'une configuration . . . 152
11.4.1 Déploiement des nœuds de fusion . . . 153
11.4.2 Déploiement d'une étape locale . . . 154
11.5 Communication . . . 154
11.5.1 Exécution du processus . . . 156
11.5.2 Contrôle du système . . . 156
11.6 Synthèse . . . 158

12 Conclusion **159**
12.1 Contributions . . . 159
12.2 Perspectives . . . 162

IV Annexes **171**

13 Premiers pas avec eZFusion **173**
13.1 Compatibilité visée . . . 173
13.1.1 Systèmes d'exploitations compatibles . . . 174
13.1.2 Infrastructure réseau . . . 174
13.2 Installation et utilisation . . . 174
13.3 Premiers pas . . . 175
13.3.1 Démarrage de eZFusion . . . 175
13.3.2 Rédaction d'une configuration . . . 177
13.3.3 Déploiement d'une configuration . . . 178
13.3.4 Contrôle de l'exécution du processus . . . 180
13.4 Administration . . . 183
13.4.1 Gérer les plates-formes OSGi . . . 183
13.4.2 Gérer la compatibilité des fonctions de fusion . . . 183
13.4.3 Gérer les attributs du système . . . 184

13.5 Développement de fonction de fusion 184
13.5.1 Développement d'une fonction de fusion 185
13.5.2 Installation et utilisation d'une fonction de fusion 186

Table des figures

1.1 Positionnement de l'utilisation de nos travaux. 23

2.1 Exemple de système réparti de fusion d'informations : application à la recherche et la localisation de restaurants. 26
2.2 Topologies des systèmes répartis de fusion d'informations [LCK+97] : les flèches matérialisent le sens des canaux de communications. 30
2.3 Modification du but pour un système de fusion d'informations. 32
2.4 Modification du traitement dans un système de fusion d'informations. 32
2.5 Modification de la mise en œuvre dans un système de fusion d'informations. . . . 33
2.6 Positionnement de la gestion de la volatilité des ressources. 34
2.7 Positionnement de l'adaptation de la puissance de calcul allouée au traitement. . 35
2.8 Positionnement du passage à l'échelle du système. 36
2.9 Positionnement de la généricité et de la réutilisation. 36
2.10 Contrôle des systèmes répartis de fusion d'informations : positionnement de nos travaux dans le repère traduit de [RNL03]. 38

3.1 Premier niveau d'étude des systèmes répartis de fusion d'informations : le matériel de mise en œuvre. 42
3.2 Deuxième niveau d'étude des systèmes répartis de fusion d'informations : les outils de mise en œuvre. 42
3.3 Positionnement de nos travaux par rapport aux solutions existantes. 45

4.1 Troisième niveau d'étude des systèmes répartis de fusion d'informations : les paradigmes de mise en œuvre. 48
4.2 Modèle de conception producteur-consommateur : les flèches représentent le sens des échanges d'informations. 49
4.3 Quatrième niveau d'étude des systèmes répartis de fusion d'informations : les modèles de fusion. 50
4.4 Cinquième niveau d'étude des systèmes répartis de fusion d'informations : la théorie de la fusion. 53

5.1 Architecture de contrôle à boucle fermée, appliquée à un système de fusion d'informations. 57
5.2 Détail de la boucle de contrôle : les flèches représentent la dépendance entre les tâches de contrôle. 60

6.1 Le processus de fusion est décrit par un graphe de flots de données. 63
6.2 Représentation d'une fonction de fusion. 65
6.3 Représentation d'une source d'informations. 65
6.4 Représentation d'un actionneur. 65
6.5 Exemple de description de processus de fusion d'informations. 67

6.6 Fusion probabiliste non bayésienne. 68
6.7 Fusion probabiliste bayésienne. 68
6.8 Décision probabiliste. 69
6.9 Fusion en théorie des possibilités : f est l'opérateur d'agrégation et d est un des éléments de l'ensemble de discernement. 69
6.10 Fusion en théorie des fonctions de croyances : m_j la fonction de masse de la source s_j, la fusion [Sme90] s'effectue pour tout les sous-ensembles $A \subset D$. 69
6.11 Organisation des éléments d'une configuration du système de fusion d'informations. 71
6.12 Ensemble des affectations possibles pour une configuration, vue étendue. 72
6.13 Ensemble des affectations possibles pour une configuration. 72
6.14 Exemple d'affectation d'éléments de fusion sur un ensemble des cadres d'exécution. 73
6.15 Exemple d'affectation d'éléments de fusion sur un ensemble des cadres d'exécution, vue étendue. 73
6.16 Exemple de configuration : affectation des éléments du processus de fusion aux ressources disponibles. 74
6.17 Tâche de contrôle liée à la configuration. 75
6.18 Organisation des éléments d'une implémentation de processus de fusion. 76
6.19 Composition d'un nœud de fusion. 77
6.20 Exemple d'interactions de nœuds de fusion pendant une exécution. 78
6.21 Exemple d'interactions de nœuds de fusion pendant une transition. 79
6.22 Tâche de contrôle liée à l'implémentation. 82
6.23 Variables de contrôle issues de l'implémentation du processus de fusion. 82
6.24 Temps d'exécution d'une entité d'exécution. 83
6.25 Taux d'utilisation d'un nœud de fusion et d'un cadre d'exécution, sur un intervalle de temps constant. 83

7.1 Traduction dans notre modèle d'analyse de l'implémentation des éléments de fusion. 89
7.2 Traduction dans notre modèle d'analyse de l'implémentation d'un lien entre deux éléments de fusion (a) en une étape locale (b) ou distante (c). 89
7.3 Traduction dans notre modèle d'analyse des ressources CPU et réseau : en grisé les éléments déjà traduits. 90
7.4 Traduction dans notre modèle d'analyse de l'implémentation d'un processus de fusion sur un cadre d'exécution : en grisé les éléments déjà traduits. 92
7.5 Traduction dans notre modèle d'analyse de l'implémentation d'un processus de fusion sur plusieurs cadres d'exécution : en grisé les éléments déjà traduits. 92
7.6 Description d'un processus de fusion puis bouclage d'analyse associé : en grisé les éléments déjà traduits. 94
7.7 Tâche de contrôle liée à la performance. 96

8.1 Le processus d'extraction de connaissance dans les données. 102
8.2 Exemple d'une séquence d'événements tiré de [MR04]. 103
8.3 Analyse des mesures du système pour le diagnostic de défaillance. 103
8.4 Tâche de contrôle liée à la robustesse. 105
8.5 Tâche de contrôle liée à la reconfiguration. 109

9.1 Les différents types de répartition : le parallélisme de contrôle sans dépendance et avec dépendance. 112
9.2 Les différents types de répartition : parallélisme de donnée et parallélisme de flux. 113
9.3 Les différents types de répartition : la réplication. 113

9.4 Structures des SMA [BD95] : (1) groupe simple, (2) équipe, (3) hiérarchie simple et (4) hiérarchie multi-niveaux. . . . 114
9.5 Tâche de contrôle hiérarchique : agent1→agent2 représente la relation « agent1 exploite les données de l'agent2 ». . . . 117
9.6 Organisation du système de contrôle (Figure 5.1) et détail de la boucle de contrôle avec les quatre tâches du système de contrôle (Figure 5.2) 118
9.7 Positionnement des tâches de contrôle au sein des systèmes de contrôle et d'exécution. . . . 118
9.8 Les tâches de contrôle et leurs niveaux dans la hiérarchie : cas des tâches *Implémentation*, *Performance* et *Robustesse*. . . . 119
9.9 Les tâches de contrôle et leurs niveaux dans la hiérarchie : cas des tâches *Implémentation*, *Configuration* et *Reconfiguration*. . . . 120
9.10 Résultat de la simulation d'automates cellulaires auto-stabilisés tiré de [LGMK05]. 124
9.11 Mécanisme de différentiation et d'auto-stabilisation appliqué à l'organisation d'une tâche de contrôle hiérarchique : a) trois groupes d'agents pour trois niveaux de contrôle adjacents, b) vue du système sans la notion de distance, et c) hiérarchie de contrôle basée sur les groupes d'agents. . . . 125
9.12 Activation d'une niveau de la hiérarchie de contrôle : coordination par division du contrôle. . . . 127
9.13 Activation d'une niveau de la hiérarchie de contrôle : coordination par synchronisation du contrôle. . . . 127
9.14 Communication au sein d'une tâche de contrôle. . . . 128

10.1 Observation de phénomènes oscillatoires. . . . 136
10.2 Observation de la répartition. . . . 136
10.3 Profils de la fonction de probabilité de Fermi-Dirac : variation du point d'inflexion et de la pente. . . . 136
10.4 Résultats de simulation, analyse A. . . . 139
10.5 Résultats de simulation, analyse B. . . . 141
10.6 Résultats de simulation, analyse C. . . . 142
10.7 Résultats de simulation, analyse D. . . . 143
10.8 Logo de eZFusion, prononcé « easy fusion ». . . . 147

11.1 Conditionnements de eZFusion dans un cadre d'installation OSGi. . . . 152
11.2 Composants et services de eZFusion dans un cadre d'installation OSGi. . . . 153
11.3 Organisation du déploiement des nœuds de fusion dans eZFusion : dans cet exemple F1 et F3 sont des sources d'informations utilisées par F2 et F4. . . . 155
11.4 eZFusion dans un cadre d'installation OSGi : mise en œuvre de deux liens, de F1 vers F2 en local et de F3 vers F4 grâce à ROSGi. . . . 157
11.5 eZFusion dans un cadre d'installation OSGi : protocoles de communication. . . . 158

12.1 État d'avancement détaille du système de contrôle. . . . 160

13.1 Logo de eZFusion, prononcé « easy fusion ». . . . 173
13.2 Liste des services OSGi importés et exposés par eZFusion. . . . 175
13.3 Interface Web de eZFusion. . . . 176
13.4 Exemple de processus de fusion : 3 sources d'information reliées à 1 actionneur. . 178
13.5 Exemple de fichier de configuration : ici trois sources d'information et un actionneur. 179
13.6 Exemple de fichier de scénario : ici le déploiement d'une configuration sur deux cadres d'exécution. . . . 180

13.7 Exemple de processus de fusion : ici un exemple liée à la domotique, un haut le processus de fusion et en bas le résultat sur lequel nous avons repéré certains des éléments du processus. 181
13.8 Interface d'information d'eZFusion : exemple de visualisation des temps de traitement (en msec) et des taux d'utilisation de nœuds de fusion. 182
13.9 Interface d'information d'eZFusion : mise à jour de la valeur d'un paramètre du nœud de fusion *clock* et visualisation du changement au niveau du temps d'exécution. 182
13.10 Diagramme de classes UML d'eZFusion : résumé de l'utilisation d'une implémentation de fonction de fusion, ici avec deux exemples. 184

Chapitre 1

Introduction générale

La fusion d'informations est aujourd'hui un domaine de l'informatique, pourtant il est possible d'en trouver des exemples dès le XVIII siècle, lorsque Diderot et d'Alembert identifient le recoupement de témoignages comme de la fusion d'informations. Les bases de cette discipline sont alors déjà définies, et l'on retrouve les notions de sources d'informations incertaines et d'informations imprécises.

Cette thèse ne porte pas sur la fusion d'informations, mais sur le contrôle des systèmes qui la mettent en œuvre. La fusion d'informations est aujourd'hui appliquée dans des domaines très variés, allant des sciences humaines aux applications militaires, dès lors que des vues partielles de la réalité doivent être combinées. C'est par exemple le cas en météorologie, où des informations fournies par les satellites et les stations au sol sont fusionnées pour obtenir la force et la position d'une perturbation. Des systèmes de fusion d'informations sont également appliqués dans des domaines plus ludiques, notamment dans les derniers logiciels téléchargeables sur les téléphones portables, où les informations provenant de plusieurs sources, comme par exemple des entrées d'un annuaire téléphonique ou une position GPS, sont fusionnées pour connaître l'adresse des restaurants les plus proches. Nous reprendrons cet exemple dans la suite du mémoire car il illustre également le caractère variable des sources d'informations connectées au système. Nous le compléterons au besoin pour illustrer certains points de nos travaux.

La fusion d'informations est un domaine de recherche à part entière. De nouvelles théories ou de nouveaux modèles sont régulièrement publiés et la fusion d'informations est aujourd'hui un domaine de recherche incontournable. Le développement de systèmes informatiques mettant en œuvre des processus de fusion est également très actif et des efforts d'ergonomie ont été réalisés pour que les chercheurs en fusion d'informations puissent les manipuler.

Les systèmes de fusion d'informations ont su évoluer ces trente dernières années, principalement sous la contrainte de nouvelles technologies. Alors qu'à la préhistoire de l'Informatique, un bâtiment dédié était nécessaire pour exécuter un système de fusion d'informations capable de gérer la télémétrie d'une fusée, l'équivalent peut aujourd'hui être concentré dans un téléphone au creux de la main. Outre cette évolution radicale de l'architecture des systèmes de fusion d'informations, les domaines applicatifs, hier cloisonnés aux laboratoires, militaires ou non, pullulent sur les réseaux informatiques publics. Bientôt, certains constructeurs en téléphonie pourraient avoir comme accroche : « si vous voulez faire de la fusion d'informations, il y a une application pour ça ».

Nos travaux se situent à deux niveaux. Ainsi, comme l'illustre la Figure 1.1, nous proposons un modèle de description et de déploiement d'un processus de fusion sur un ensemble de ressources réparties. La description du processus, fournie par l'expert en fusion, est indépendante des ressources nécessaires à son exécution. L'implémentation du processus de fusion, c'est à dire le code exécutable qui effectue le traitement, provient du développeur du système, indépendamment

des mécanismes de répartition et de contrôle du système.

Le système que nous proposons combine la description du processus et son implémentation, et configure le système réparti pour que les ressources disponibles appliquent effectivement le traitement des informations. L'exécution du processus de fusion, et l'éventuelle reconfiguration du système réparti, s'effectue de façon transparente pour l'utilisateur final.

Pour réaliser ce système, nous fournissons un cadre d'implémentation qui simplifie la traduction d'une opération de fusion dans un code exécutable. De plus, notre solution propose également le contrôle autonome du système réparti de fusion d'informations, afin de minimiser les interventions de l'administrateur du système. Ce contrôle permet l'évaluation des performances d'une configuration du système avant son application, mais aussi le suivi de ces performances pendant l'exécution. Cette surveillance du système pendant l'exécution permet également d'en améliorer ses performances lorsqu'une meilleure configuration est trouvée (par exemple lors de l'ajout de nouvelles ressources). Notre système de contrôle est ainsi divisé en cinq tâches :

- la recherche d'une configuration,
- l'analyse des performances d'une configuration,
- la mise en œuvre d'une configuration,
- la surveillance de l'exécution du processus de fusion,
- le déclenchement d'une reconfiguration du système.

Chacune de ces tâches de contrôle soulève des problèmes que nous proposons de régler dans le cadre des systèmes répartis de fusion d'informations. L'analyse des performances est par exemple basée sur une traduction et une évaluation automatique d'une configuration dans le modèle des réseaux de Petri stochastiques généralisés.

Enfin, notre système de contrôle est également réparti sur l'ensemble des ressources du système. En effet, dans un souci de performance et de robustesse, nous avons décidé de décentraliser le système de contrôle en répartissant chaque tâche sur l'ensemble des ressources. Ainsi, grâce à une approche bio-inspirée, chaque élément du système active une partie du système de contrôle selon sa puissance de calcul et sa présence dans le système. Un élément puissant et permanent, par exemple un ordinateur de bureau, aura alors en charge une part plus importante du contrôle, à l'inverse d'un élément peu puissant, comme un téléphone portable.

Problématiques De nos jours, les problématiques des systèmes de fusion d'informations sont en partie guidées par les évolutions technologiques. Ainsi, les systèmes de fusion sont intrinsèquement répartis : il n'est pas rare que les sources d'informations, les actionneurs et les ressources de calcul ne soient pas dans la même pièce. La disponibilité de ces mêmes ressources, qu'elles fournissent des informations ou les traitent, est par ailleurs sujette à des fluctuations : d'autres programmes exécutés sur un ordinateur peuvent par exemple modifier la puissance de calcul disponible. En outre, la présence d'un capteur, d'un écran, ou plus généralement d'un nœud du système peut également être intermittente - un ordinateur portable peut changer de pièce. Enfin, devant la complexité des systèmes logiciels sous-jacents, les développeurs de systèmes répartis réutilisent au maximum les outils déjà diffusés.

La difficulté, relevée par la communauté, a été de proposer des systèmes capables de déployer un processus de fusion donné sur un tel ensemble de ressources réparties. De nombreux travaux, largement éprouvés et aujourd'hui matures, ont été proposés dans les domaines des capteurs intelligents et des nuages de capteurs. Les processus mis en œuvre ne permettent pourtant pas le traitement de données volumineuses, principalement car les puissances de calcul embarquées sont limitées.

Le type de systèmes que nous visons, ne dispose toujours pas d'outils paradigmatiques suffisamment adaptatifs pour gérer la dynamicité des ressources disponibles, mais également assez simples pour être réutilisés par les communautés d'experts et de développeurs. De plus, les outils

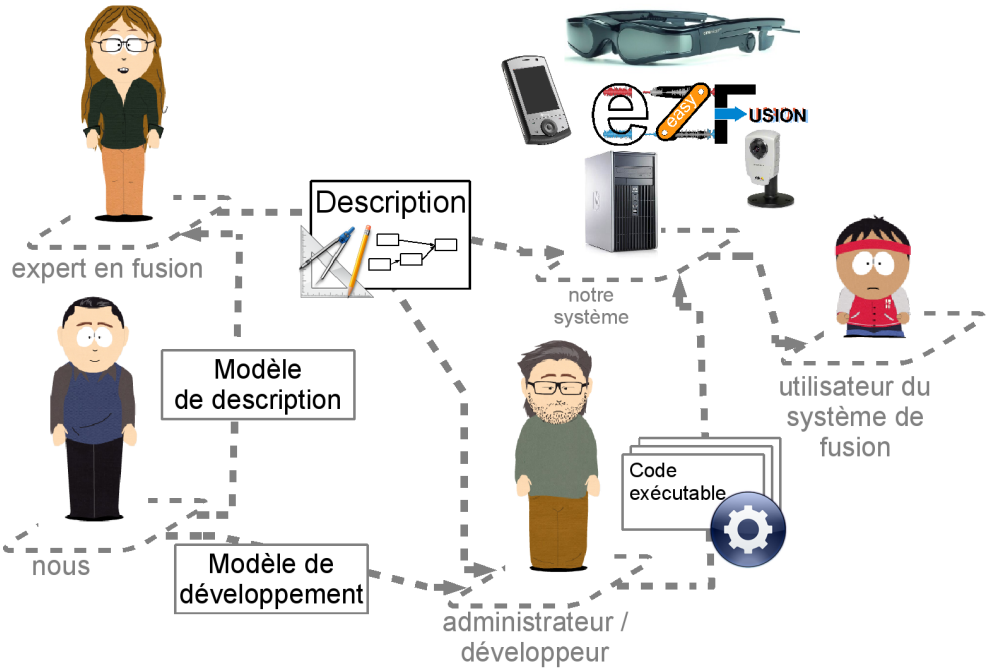


FIG. 1.1 – Positionnement de l'utilisation de nos travaux.

de mise en œuvre de système de fusion d'informations n'assurent pas, le plus souvent, le contrôle du processus de fusion lors de son exécution. En effet, pour un processus de fusion donné, il peut être nécessaire d'adapter certains paramètres du traitement, pendant l'exécution, sans reconfigurer l'intégralité du système.

Outre le manque de contrôle du processus de fusion et la gestion limitée de la dynamicité des ressources, les systèmes de fusion d'informations doivent également relever d'autres défis que nous ne traitons pas dans cette thèse. Le premier, incontournable dès lors que des communications interviennent, repose sur la confidentialité des données échangées dans le système. Ainsi, que les données soient liées au contrôle du système ou aux résultats partiels transmis lors du traitement, le domaine de la sécurité des systèmes informatiques regorge de protocoles et de mécanismes qui préviennent d'un détournement technique. La planification de l'exécution des traitements élémentaires d'un processus de fusion est également un domaine qui contribue à l'amélioration des systèmes de fusion d'informations. Les problèmes abordés concernent également la disponibilité des ressources, mais les solutions sont proposées selon d'autres méthodes. La gestion des niveaux énergétiques ou des fenêtres de transmission de données en sont deux exemples récurrents.

Objectifs Nos objectifs ont progressivement évolué pour finalement se fixer autour de plusieurs axes :

- comment décrire et exécuter un processus de fusion d'informations ?
- quelle est la nature du contrôle d'un système réparti de fusion d'informations et comment le mettre en œuvre ?
- comment gérer l'évolution de l'ensemble des ressources disponibles et leur nature ?
- comment répartir le système sur un ensemble de ressources donné ?

Ces objectifs, une fois identifiés, nous ont guidés pendant ces trois années de recherche. Les phases de développement expérimental et de travail théorique se sont succédées et ce mémoire reprend la substance issue de nos réflexions.

Les contributions que nous proposons sont donc :
- un modèle de contrôle des systèmes répartis de fusion d'informations ;
- un modèle bio-inspiré de répartition du système contrôle ;
- un modèle de description et de déploiement des processus de fusion dans un contexte réparti ;
- un modèle d'analyse de performances d'un système réparti de fusion d'informations ;
- et une application des résultats dans un outil logiciel.

Plan du mémoire Ce mémoire détaille les points précédents dans trois parties. Ainsi, à la suite de la présentation détaillée de la problématique de cette thèse dans le Chapitre 2, la partie « État de l'art » regroupe notre étude de l'existant. Cette partie est composée de deux chapitres qui traitent respectivement des systèmes de fusion d'informations tels qu'ils sont exploités dans la communauté (Chapitre 3), et des modèles de fusion d'informations manipulés par les concepteurs de processus de fusion (Chapitre 4).

La seconde partie présente le cœur de ce mémoire. La partie « Travaux théoriques » formalise les objectifs des cette thèse au regard des systèmes existants (Chapitre 2.2.2), puis détaille le système de contrôle que nous proposons. Ce système est présenté selon trois aspects selon que nous nous intéressons à sa description globale, dans le Chapitre 5, au mécanisme de répartition que nous proposons (Chapitre 9) ou aux détails des tâches de contrôle qui le compose (Chapitres 6 et 7). Certaines de ces tâches sont introduites dans le Chapitre 8 où nous regroupons des travaux prospectifs que nous approfondirons dans de futures recherches.

Les pistes que nous avons suivies lors du développement logiciel de notre système sont présentées dans la dernière partie de ce mémoire. La partie « Mise en œuvre » regroupe les différentes directions qui s'offraient à nous, et nous les expliquons en évitant les travers d'une documentation technique.

Ce mémoire est finalement clos par un rappel de nos objectifs ainsi que par une mise en perspective personnelle de nos résultats. Nos perspectives de recherche terminent ce dernier bilan.

Chapitre 2

Problématique et objectifs de la thèse

Ce chapitre présente dans un premier temps notre vision de la fusion d'informations et des systèmes qui la mettent en œuvre, qu'ils soient répartis ou non. La seconde partie de ce chapitre approfondit les objectifs que nous avons dressés dans l'introduction du mémoire. Nous donnons l'architecture des systèmes que nous visons puis détaillons chaque objectif dans le cadre des systèmes répartis de fusion d'informations. Nous positionnons alors notre approche selon un découpage fonctionnel du contrôle du système mais aussi selon différents niveaux d'abstraction.

2.1 Systèmes de fusion d'informations

L'exemple que nous proposons dans ce paragraphe nous permet d'illustrer notre vision des Systèmes de Fusion d'Informations (SFI). Un rappel historique dresse les origines des SFI électroniques, puis nous proposons une liste des évolutions majeures de ces systèmes et des nouvelles problématiques qui en résultent. Il n'y a pas de consensus sur la définition d'un SFI, aussi avons-nous choisi d'en sélectionner quelques-unes pour donner un aperçu de l'éventail des points de vue. Nous identifions la définition la plus proche de notre étude et donnons notre définition des SFI en conclusion de ce paragraphe.

2.1.1 Exemple de fusion d'informations

La fusion d'informations est appliquée tous les jours dans des domaines très variés. Le cerveau humain effectue par exemple de la fusion d'informations en permanence pour évaluer la position d'un objet ou la nature d'un danger.

Notre exemple, illustré par la Figure 2.1, montre un processus de fusion d'informations où un utilisateur demande l'adresse des restaurants les plus proches. Son PDA fournit les coordonnées GPS au système de fusion qui interroge trois sources d'informations. La première est connectée à une base de données et fournit les adresses des restaurants. Les deux autres sources d'informations produisent deux vues de la région où se situe l'utilisateur. Chaque vue fournit une partie du résultat final. Une vue aérienne prise par un satellite dans le domaine du visible [Goo09] aidera l'utilisateur à se repérer alors qu'une seconde vue (une image radar) renseigne sur la topologie de la région de l'utilisateur et permet de classer les restaurants selon le temps de trajet à pied. Ces trois informations de base sont traitées par le système, lors de l'alignement des vues, de la projection des adresses sur celles-ci et du calcul des temps de trajet. Le résultat de la fusion est finalement transmis au PDA de l'utilisateur.

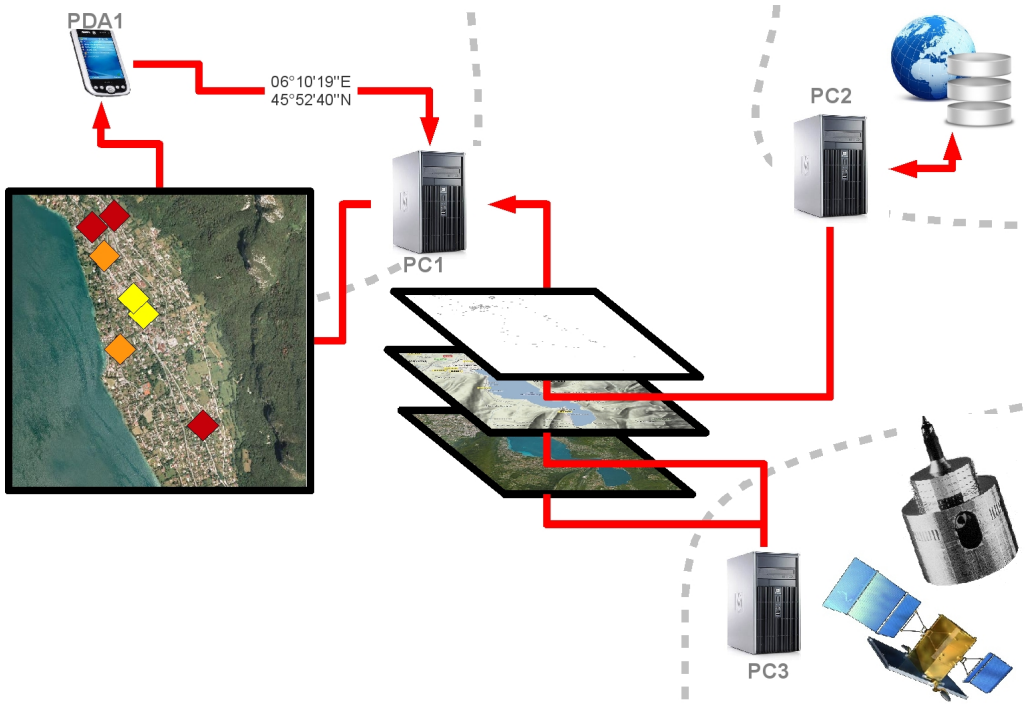


FIG. 2.1 – Exemple de système réparti de fusion d'informations : application à la recherche et la localisation de restaurants.

2.1.2 Origine des systèmes de fusion d'informations

Les premiers SFI électroniques sont apparus pendant la seconde guerre mondiale alors que les radars et les sonars connaissaient un véritable essor. Les SFI avaient à l'époque une double utilisation. Ils amélioraient un signal, comme par exemple en limitant le bruit d'un signal reçu par un capteur, et fusionnaient des signaux provenant de deux capteurs différents pour calculer une information plus précise, comme par exemple dans des problèmes de localisation d'objets. Ces deux exemples de traitement illustrent les deux axes d'utilisation des SFI : l'amélioration d'une information par un traitement et la production d'une nouvelle information à partir des informations initiales. Une étape de prise de décision, comme le choix de l'objet identifié, suit généralement l'exécution du processus de fusion en en consommant les résultats.

2.1.3 Évolutions des systèmes de fusion d'informations

Les évolutions des SFI se sont effectuées à plusieurs niveaux : au niveau des sources d'informations et au niveau des unités de traitement, qu'elles soient électroniques ou logicielles.

L'amélioration de la précision des capteurs, ainsi que leur miniaturisation, a été une première évolution des SFI. De là, ont découlé de nouvelles problématiques comme l'augmentation des données, tant par leur taille que par leur quantité. L'augmentation du nombre de messages transmis s'est effectuée conjointement à l'évolution du nombre de sources. Cette augmentation des données transmises a également eu lieu dans les niveaux de traitement, notamment lors de la répartition du processus de fusion où les échanges de résultats partiels se sont ajoutés aux informations capturées et produites par le système. L'évolution des sources d'informations logicielles s'est faite en parallèle de celles des sources électroniques. Quasiment inexistantes dans les années 1970, elles sont maintenant omniprésentes : prenons l'exemple des SFI basés sur Internet

où l'utilisation d'un service Web ou d'une base de données est aujourd'hui courante [Yao05].

Une autre évolution majeure réside dans la hausse des performances des unités de traitement. Celles-ci sont généralement électroniques, i.e. un processeur dédié, quand le SFI dispose de ressources limitées et logicielles dans les autres cas. Dans le cas d'un SFI embarqué, comme sur un drone [KZK97], les éléments de traitement sont encore majoritairement électroniques même si leur part tend à diminuer, principalement grâce à la miniaturisation des architectures de type PC. Ce type d'architecture s'étant démocratisé, il existe maintenant pléthore d'éléments de traitement logiciels comme des filtres ou des fonctions de compositions. L'évolution des unités de traitement s'est par ailleurs faite au niveau des architectures utilisées. Alors qu'il y a quelques années un SFI nécessitait un ordinateur dédié, les SFI actuels s'exécutent sur un ensemble de machines pouvant aller du simple réseau domestique au nuage de PC.

Ce tour d'horizon des évolutions des SFI se termine par un point sur la nature des mécanismes de fusion. En effet, les processus de fusion d'informations sont à présent mis en œuvre par des méthodes aussi bien abstraites qu'expérimentales. C'est par exemple le cas de certains réseaux neuronaux [PC03] où les fonctions implémentées sont paramétrées par des coefficients dont la valeur est fixée suite à une phase d'expérimentation. Au contraire, certains modèles [KTW04] traitent de la fusion d'informations indépendamment de l'application visée, voire même de la méthode de fusion implémentée. Le paragraphe 4.2 dresse un panorama de ces modèles de fusion d'informations.

2.1.4 Nouvelles problématiques des systèmes de fusion d'informations

Pour illustrer les nouveaux défis que les SFI doivent relever, citons [Lli07] où l'auteur expose les « objectifs de la fusion d'informations » dans le cadre de conflits internationaux. Différents objectifs stratégiques sont avancés comme la localisation de cyber-terroristes dans le cyberespace ou l'interaction nécessaire des infrastructures informatiques civiles et militaires. La liste de ces objectifs est suivie de cinq autres objectifs technologiques pertinents.

L'auteur utilise le terme de « fusion d'informations » que nous traduisons par « processus de fusion » ou « système de fusion » selon notre interprétation du niveau conceptuel visé. Voici la liste des objectifs technologiques proposés :

- la robustesse et le passage à l'échelle du système dans le but d'étendre le cadre opérationnel de la fusion d'informations ;
- la conception de processus de fusion d'informations qui fonctionnent effectivement et efficacement dans des réseaux informatiques hostiles ;
- la prise en compte, lors de la conception des processus de fusion, des cascades d'effets causés par le système sur le monde extérieur ;
- la conception de processus de fusion d'informations multi-perspectives qui exploitent des concepts définis selon le réseau opérationnel (un même concept n'a pas la même définition si l'on considère un soldat ou un terroriste) ;
- la conception de processus de fusion d'informations qui supportent un plus large spectre de types d'informations.

Certains de ces objectifs sortent du cadre de notre étude. Les paragraphes suivants introduisent les objectifs des futurs SFI que nous avons retenus et le Chapitre 2.2.2 les détaille dans le cadre de notre approche.

De nombreux capteurs volatiles Il paraît aujourd'hui clair que les futurs SFI doivent être en mesure de s'adapter aux capteurs disponibles à deux niveaux.

Premièrement, le type et le nombre des capteurs installés dans un SFI ne sont *a priori* pas connus du concepteur du système. La découverte des éléments disponibles dans le SFI, avant le

traitement des informations, est un prérequis des SFI actuellement accepté par la communauté.

Deuxièmement, la gestion de la persistance d'un capteur et de l'ajout de nouveaux éléments, notamment de communication, doit également entrer dans les caractéristiques des SFI. Il est ainsi convenu qu'une augmentation des éléments de traitement dans le système peut entraîner une dégradation de la fiabilité de ce même système. Les SFI doivent donc gérer les défaillances de tout ou partie des éléments qui le constituent. En outre, à l'opposé de la perte d'un capteur, le système doit être en mesure d'intégrer efficacement de nouvelles sources d'informations.

Des ressources limitées et plus de données à traiter Les évolutions des SFI ont été menées dans deux directions opposées. Il en est ainsi de la recherche de systèmes toujours plus économes en ressources, mais également toujours plus puissants pour traiter rapidement des données plus volumineuses. Ce nouveau challenge des SFI contraint les futurs systèmes à adapter la puissance de calcul allouée aux traitements appliqués aux informations. Cette adaptation suit d'une part le processus de fusion déployé, duquel le système évalue la puissance de calcul nécessaire, et suit d'autre part les éléments d'exécution effectivement disponibles. Une adaptation facile, voire automatique, des SFI est une des propriétés indispensables des futurs systèmes proposés.

De nouvelles méthodes de fusion sans systèmes adaptables En parallèle des évolutions techniques, la communauté scientifique a également élaboré de nouveaux modèles, ou méthodes, de fusion d'informations. Leurs tests et leurs applications pratiques sont le plus souvent basés sur un système développé sans objectif d'adaptation ou de généralisation. Il en résulte que la communauté scientifique a produit en abondance des systèmes mettant en œuvre les résultats d'études isolées. Cette confusion entre méthodes de fusion et outils de mise en œuvre est malheureuse car deux catégories d'outils sont alors proposés. Les premiers, issus de développements isolés, ont pour but l'exécution d'un processus de fusion d'informations donné. Le développement des seconds est au contraire guidé par la généricité du système, afin d'être compatible avec plusieurs modèles de fusion. Ces outils sont de plus en plus faciles d'accès et réutilisables, or aucun n'émerge significativement. Selon nous, une des raisons possibles de cet isolement des outils est l'effort de prise en main qui reste sans doute trop important.

Notre approche propose donc une autre voie où, contrairement à une simplification du développement en direction de l'expert en fusion, nous proposons la distinction du rôle de concepteur de processus de fusion, de celui de développeur. Le concepteur décrit le processus de fusion à mettre en œuvre et le développeur écrit le code exécutable du traitement à appliquer, indépendamment des mécanismes de communication et de contrôle. Cette proposition, déjà éprouvée dans d'autres domaines de l'informatique comme la conception Web [New01], est prometteuse et devrait s'accompagner de la démocratisation des futurs outils qui seront, cette fois, réutilisés.

2.2 Définitions

L'étude bibliographique menée dans cette thèse nous a fait rencontrer de nombreuses définitions de « la fusion d'informations ». Nous en avons sélectionné quelques-unes qui illustrent les différentes approches possibles.

Pour le néophyte, [Edi09] propose la définition suivante :
« Fusion, nom, féminin : (...) Réunion en un seul groupe de divers éléments distincts. »

Les spécialistes du domaine ont pour leur part étudié la fusion d'informations selon différents points de vue. [Blo05] s'intéresse au processus de fusion :
« La fusion d'informations consiste à combiner des informations hétérogènes issues de plusieurs sources afin d'améliorer la prise de décision. »

[SBW98] souligne le but de la fusion d'informations :
« La fusion d'informations est la combinaison de données dans le but de raffiner l'estimation ou la prédiction d'un état. »

[KTW04] reformule l'enchaînement des étapes de fusion d'informations puis de décision :
« Les systèmes de fusion de décisions peuvent être vus comme des systèmes de fusion d'informations. La fusion d'informations est ainsi un moyen d'implémenter un système de fusion de décisions. »

Enfin [BDPP06] détaille les méthodes de fusion et leurs opérateurs :
« Le but de la fusion d'informations est l'extraction de ce qui est vrai au milieu d'une collection de données souvent imprécises et contradictoires. »

2.2.1 Définition d'un système de fusion d'informations

Les définitions précédentes couvrent le large éventail des SFI actuellement déployés. Ces définitions, souvent générales, ne sont précisées que lors de la mise en œuvre des applications des SFI. Une unique définition des SFI est donc difficile à élaborer car ces systèmes peuvent être étudiés sous plusieurs aspects, les Chapitres 3 et 4 en dressent un état de l'existant. Dans le cadre de cette thèse, nos travaux se focalisent sur la description du processus de fusion sous la forme d'un graphe ainsi que sur la mise en œuvre et le contrôle de ce processus sur un ensemble de ressources réparties. Nous détaillons ces points dans la suite du mémoire et nous proposons la définition suivante d'un SFI :

Définition 2.2.1. *Un système de fusion d'informations (SFI) est un système matériel et/ou logiciel qui consomme des informations produites par des sources d'informations et fournit des résultats de meilleure qualité, ou de nature différente, que les informations de base.*

Cette définition, proche de celle présentée dans [Blo05], découle de l'étude que nous souhaitons faire des SFI. En effet, nous nous intéressons autant aux éléments physiques, ou électroniques, qu'à ceux logiciels. Les capteurs sont à l'heure actuelle majoritairement physiques, principalement parce que les SFI sont appliqués dans des domaines eux aussi physiques (dans le sens de concrets et de tangibles). La nature des éléments des SFI, électronique ou logicielle, tend à changer comme nous l'expliquons dans la Paragraphe 2.1.3 et la prépondérance d'un type d'éléments de traitement est à présent beaucoup moins marquée. Ainsi, les opérations de fusion sont mises en œuvre aussi bien par des éléments électroniques que logiciels, tant au niveau des sources d'informations que lors du traitement. Les résultats issus de ces traitements sont d'une part de meilleure qualité lorsque les SFI en améliorent la précision ou en réduisent l'incertitude, d'autre part de nature différente comme par exemple lors d'une caractérisation ou d'une identification.

2.2.2 Systèmes répartis de fusion d'informations

Définition 2.2.2. *Notre approche pose qu'un Système Réparti de Fusion d'Informations (SRFI) est un SFI dont les sources, les actionneurs ou les éléments de traitement ne disposent pas d'une mémoire partagée et nécessitent la communication d'informations par un élément indépendant du système.*

Prenons comme exemple un système composé de deux PC reliés en *ethernet*, l'un connecté à une caméra et l'autre connecté à un écran. Ce système définit un SRFI lorsque un filtre est appliqué à l'image provenant de la caméra puis que le résultat est transmis et affiché sur le second PC.

Au contraire, un système composé de plusieurs capteurs connectés directement au même PC ne définit pas un SRFI, même s'il y a un échange d'informations. Il en est de même de la communication de résultats de traitements intermédiaires entre plusieurs éléments logiciels au sein

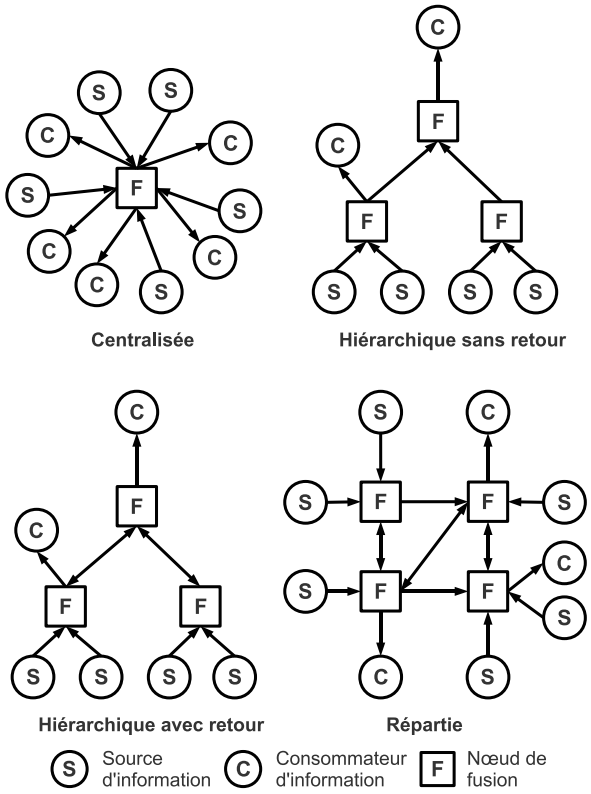


FIG. 2.2 – Topologies des systèmes répartis de fusion d'informations [LCK+97] : les flèches matérialisent le sens des canaux de communications.

du même ordinateur (hors architectures exotiques où plusieurs machines virtuelles s'exécutent sur le même PC).

L'amalgame entre un système réparti de fusion et un processus de fusion distribuée est facile, et nous devons positionner notre approche par rapport à ces deux notions. Pour cela, référons-nous à la Figure 2.2 qui illustre les différentes topologies de SRFI : les quatre graphes définissent un SRFI au sens de la Définition 2.2.2.

Ainsi, la première topologie décrit un SRFI mettant en œuvre un processus de fusion centralisée. Ce type de processus de fusion, dont le traitement est indivisible, est d'une part simple à décrire, mais implique d'autre part une convergence des communications et un regroupement des ressources. Un SRFI de ce type est adapté dans les applications *adhoc* où le réseau de communication et les puissances de traitement sont dimensionnées en fonction du processus déployé.

Les SRFI de type hiérarchique sans retour sont adaptés aux processus de fusion multi-niveaux et séquentiels, comme ceux définis dans le JDL [SBW98]. Chaque niveau représente alors un niveau de raffinement de l'information, ou un niveau sémantique. En outre, cette topologie décrit le paradigme de producteur-consommateur, comme nous le présentons dans le Chapitre 4. L'avantage d'un tel système réside dans sa répartition intrinsèque. En plus d'une concentration des communications par étage, la charge de calcul nécessaire est répartie sur l'ensemble du système.

Les deux derniers types de SRFI définissent des échanges à deux sens entre les noeuds de fusion. La nature des informations ici transmises diffère de celle des deux premières topologies. En effet, un système hiérarchique avec retour est pratique lors de la définition d'un système de contrôle, où les éléments d'un même niveau reçoivent des ordres du niveau supérieur. En outre, ce type de système ne permet pas l'échange d'informations au sein d'un même niveau. La nature des échanges d'informations dans une topologie de type « fusion distribuée », s'apparente à un dialogue entre les noeuds de fusion. C'est par exemple le cas lorsque le processus de fusion met en œuvre une étape de négociation, comme ceux rencontrés dans la fusion d'informations par des systèmes multi-agents. La difficulté d'utilisation de ce type de SRFI provient alors de la description du protocole de communication entre les noeuds de fusion.

Architecture des systèmes visés Nos travaux et leurs résultats ont été menés pour un certain cadre applicatif. En l'occurrence, nous visons les systèmes composés d'éléments disponibles au grand public comme un PDA, un *smart-phone*, une *box* Internet, un *media-center*, un PC portable ou de bureau et un serveur PC ou en lame (nous concédons que ces deux dernières architectures sont plus destinées à une utilisation professionnelle).

Les architectures plus petites, comme les nœuds de capteurs ou les capteurs intelligents, ainsi que les architectures plus larges, comme les grappes de serveurs ou les grilles de calcul, sortent du cadre applicatif de nos travaux. Cette restriction du domaine d'étude découle de l'analyse des objectifs posés dans chaque communauté. En effet, les problématiques discuter dans le domaine des capteurs poussière ou des motes s'orientent autour de la gestion des ressources embarquées et du routage des communications dans le réseau de capteurs. Ce domaine s'apparente également au contrôle du capteur, néanmoins nous écartons de nos objectifs la gestion de la quantité d'énergie disponible ou la puissance d'émission d'un message. La limite haute de l'architecture des systèmes dont nous visons le contrôle dérive également de la modification des objectifs du contrôle. Ainsi, l'amélioration des performances dans une grille de calcul se traduit en partie par l'équilibrage de la charge de calcul entre les éléments du système. En outre, la présence de ces éléments dans le système est stable et le besoin de reconfiguration est alors moins marqué.

Outre le périmètre des systèmes visés, l'implémentation de nos résultats suppose la disponibilité d'une machine virtuelle Java et d'un canal de communication TCP entre les nœuds du système. Ce choix justifie également l'écartement des « petites » architectures, du moins pour les quelques années à venir. Nous ne discutons de ces choix dans ce chapitre et préférons les présenter

FIG. 2.3 – Modification du but pour un système de fusion d'informations.

FIG. 2.4 – Modification du traitement dans un système de fusion d'informations.

dans la partie « Mise en œuvre » du mémoire. Concernant les communications entre ces nœuds, les problèmes de sécurité, de confidentialité et d'intégrité des données sortent du cadre de nos recherches.

Synthèse Comme nous venons de le voir, les systèmes répartis de fusion d'informations peuvent prendre forme sous différentes architectures. Celles-ci s'accompagnent de leurs problématiques et nous souhaitons concentrer nos travaux sur la gestion de ressources dont la présence et la quantité sont variables, et le contrôle autonome du système avec l'adaptation de la configuration du système en fonction des besoins en calcul d'un processus de fusion. De plus, nous situons notre approche dans un contexte où l'utilisation des ressources est partagée avec d'autres applications.

Les paragraphes 2.3 à 2.6 détaillent chacun de nos objectifs et positionnent notre système vis-à-vis de la classification proposée dans [RNL03] dont nous avons traduit les trois axes de classification :

- L'axe « fonctions des réseaux de capteurs » présente cinq divisions selon les fonctions présentent sur les capteurs : la communication, le traitement, la capture, l'administration et la configuration.
- L'axe « fonctions de contrôle » est également divisé en cinq niveaux et illustre des attributs gérés par le système de contrôle, pour une fonction du réseau de capteurs donnée : la confidentialité, la sécurité, la performance, les défaillances et la personnalisation.
- Le dernier axe « niveaux de contrôle » présente à quels niveaux sont disponibles les fonctions de contrôle implémentées dans le système. Ici encore l'axe est divisé en cinq niveaux : le nœud, le nœud d'un réseau, le réseau de nœuds, le service et le métier.

2.3 Gestion de la volatilité des ressources

Degrés de dynamicité

En prélude à la gestion de la volatilité des ressources, nous dressons ici les degrés de dynamicité que nous avons rencontrés lors de notre étude. Les modifications appliquées, qu'elles soient subies ou provoquées, sont de trois natures : la modification du but du système, la modification du traitement à appliquer aux informations et la modification de la mise en œuvre de ce traitement.

FIG. 2.5 – Modification de la mise en œuvre dans un système de fusion d'informations.

Modification du but

Définition 2.3.1 (Dynamicité du but du SFI). *Une modification du but du processus de fusion implique la production d'un nouveau résultat à l'issue du traitement. Ce degré de dynamicité est activé lorsque l'accomplissement du but initial est impossible ou invalide.*

Ce type de modification peut avoir lieu lorsqu'un système analyse une situation et propose une nouvelle solution au problème posé. La Figure 2.3 illustre une modification du but où le résultat du processus est initialement appliqué par l'actionneur a_1 puis par l'actionneur a_2. Cette modification implique généralement une adaptation des traitements intermédiaires du processus de fusion que nous n'avons pas présentée sur cette illustration pour des raisons de clarté.

Le passage d'un processus de fusion dont le résultat est une vidéo (audio plus vidéo) à audio uniquement est un exemple de modification du but du processus de fusion. Un tel changement peut être déclenché suite à un ordre de l'utilisateur.

Modification du traitement

Définition 2.3.2 (Dynamicité du traitement du SFI). *Une modification du traitement est un changement du processus de fusion à but constant. Elle a lieu quand un des éléments de traitement est indisponible ou quand la combinaison de nouveaux éléments fournit un meilleur résultat.*

Par exemple, la Figure 2.4 présente deux processus de fusion équivalents du point de vue du résultat final. Néanmoins, le traitement appliqué est différent car les fonctions $f1$, $f2$ et $f3$ sont remplacées par d'autres fonctions de fusion ($f4$ et $f5$). Une modification dans la composition de filtres dans un traitement vidéo est une application courante de ce degré de dynamicité.

Modification de la mise en œuvre

Définition 2.3.3 (Dynamicité de la configuration du SFI). *Une modification de la mise en œuvre est une reconfiguration des éléments de traitement à processus de fusion constant. Elle s'applique sur le choix du code exécutable qui implémente un traitement sur l'information, et sur le choix de l'affectation d'un tel code sur les éléments physiques du système.*

Le choix d'une implémentation peut être guidé par la qualité du traitement appliqué ou par la performance d'un code exécutable, alors que le choix d'une affectation peut être basé sur la disponibilité ou la puissance d'une ressource.

La Figure 2.5 présente ces deux types de modification. La source d'information $s2$ est déplacée car la caméra initialement affectée est défectueuse et la fonction $f1$ est mise à jour par la fonction $f1'$, équivalente du point de vue du traitement, mais plus performante.

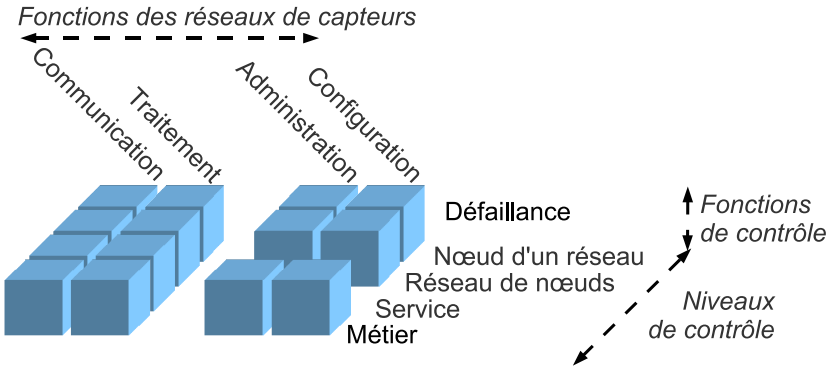


FIG. 2.6 – Positionnement de la gestion de la volatilité des ressources.

Synthèse et positionnement

Définition 2.3.4 (Ressource du SFI). *Une ressource est un élément nécessaire à la mise en œuvre d'un processus de fusion, elle peut être discrète ou continue.*

La disponibilité des ressources est *a priori* inconnue au moment de la conception du processus de fusion et nous nous positionnons dans un contexte où les ressources peuvent être partagées avec d'autres applications ou d'autres utilisateurs. Nous proposons donc une gestion des fluctuations de la disponibilité de ces ressources.

Un capteur ou un code exécutable implémentant une partie du processus de fusion, sont deux exemples de ressources « discrètes » et la puissance de calcul disponible sur un nœud du système ou la taille de la bande passante disponible sur un canal de communication illustrent bien le second type de ressources « continues ».

Synthèse Parmi les degrés de dynamicité présentés, la modification de la mise en œuvre correspond à la gestion de la disponibilité des ressources. L'ajout d'une ressource implique une éventuelle amélioration des performances du système alors que la suppression s'accompagne, dans certains cas, de la reconfiguration du système dans le but de maintenir l'exécution du processus de fusion. Afin de gérer ce degré de dynamicité, notre approche est basée sur plusieurs tâches de contrôle dédiées à la détection d'erreurs ou de défaillances, au déclenchement d'une reconfiguration puis enfin à la gestion de la transition vers une nouvelle configuration.

Positionnement de nos travaux La gestion de la volatilité des ressources, positionnée sur la Figure 2.6, est réalisée par un contrôle des défaillances, avec un évitement si possible. Notre approche propose des solutions aux points suivants :

- la détection d'une défaillance des communications entre deux nœuds du système et la reconfiguration automatique du système ; l'étape de décision de cette boucle de contrôle peut être sous-traitée à un autre système *via* un service exposé.
- la détection d'une défaillance d'un nœud du système lors du traitement d'une information et la reconfiguration automatique du système ; l'étape de décision de cette boucle de contrôle peut être sous-traitée à un autre système *via* un service exposé.

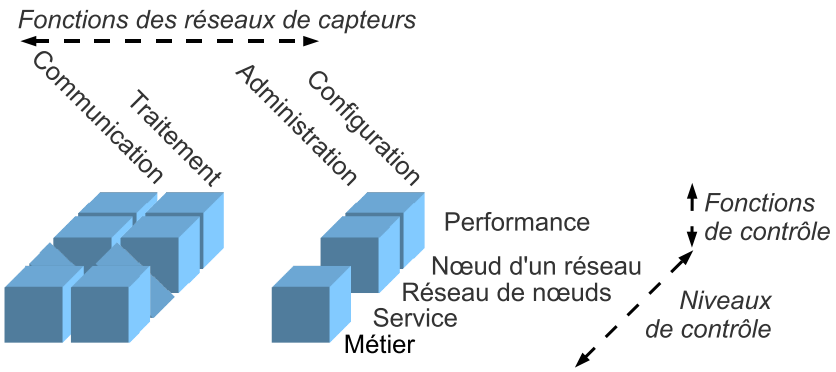


FIG. 2.7 – Positionnement de l'adaptation de la puissance de calcul allouée au traitement.

2.4 Adaptation de la puissance de calcul allouée au traitement

Il est clair que les besoins en terme de puissance de calcul ne sont pas équivalents d'un processus de fusion à l'autre. Un des objectifs de cette thèse est donc le contrôle de la puissance allouée à l'exécution d'un processus de fusion, notamment par le caractère réparti du système. La modification de la quantité de données à traiter sur un élément du système peut impliquer des problèmes dans une autre partie du système, comme par exemple la saturation des zones de stockage des résultats intermédiaires et nous proposons une gestion judicieuse de la quantité de travail affectée à chaque nœud du système. Ainsi nous proposons d'équilibrer la charge de travail demandée à chaque nœud du système grâce à des changements de configuration. Cette quantité de travail est établie selon des critères d'analyse de performance et de robustesse calculés par le système de contrôle, pendant l'exécution.

Positionnement de nos travaux La gestion de l'adaptation de la puissance de calcul allouée au traitement, illustrée par la Figure 2.7, est réalisée par le contrôle des performances du système. Notre approche propose des solutions aux points suivants :

- la surveillance et l'adaptation automatique de la quantité de données échangées entre deux nœuds du système et sur tout le réseau ; cette surveillance peut être effectuée par un autre système *via* un service.
- la surveillance et l'adaptation automatique de la charge processeur sur l'ensemble du système ; cette surveillance peut être effectuée par un autre système *via* un service.
- l'évaluation *a priori* des performances d'une configuration.

2.5 Passage à l'échelle du système

Le passage à l'échelle du système peut s'effectuer dans plusieurs dimensions que nous souhaitons gérer en partie. Ainsi, l'augmentation de la quantité de ressources disponibles doit non seulement être gérée, mais le système de contrôle doit être en mesure d'en tirer parti. Notre système est composé de deux sous-systèmes : l'un pour l'exécution du processus de fusion, l'autre pour le contrôle du premier. Ces deux systèmes sont répartis sur l'ensemble du SFI et le passage à l'échelle est évalué sur le nombre de nœuds utilisés pour l'exécution du processus de fusion, mais aussi sur la capacité du système de contrôle à gérer un SFI composé de beaucoup de nœuds (une centaine de nœuds décrits dans le Paragraphe « Architecture des systèmes visés »).

Le passage à l'échelle de certaines dimensions sortent du cadre de notre étude. Ainsi, notre

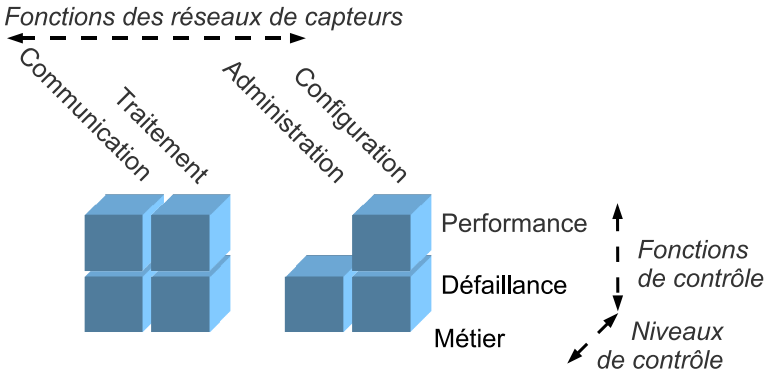


FIG. 2.8 – Positionnement du passage à l'échelle du système.

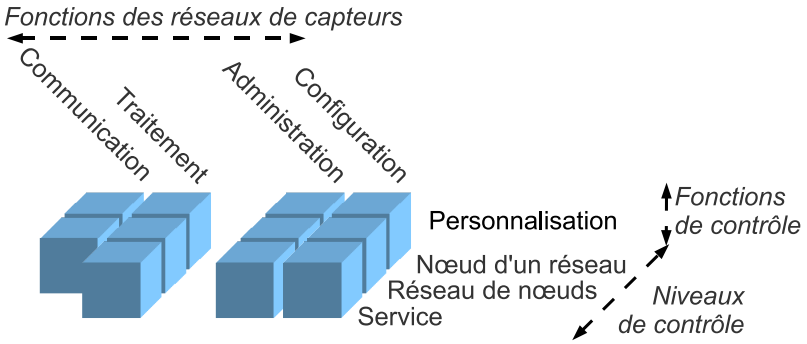


FIG. 2.9 – Positionnement de la généricité et de la réutilisation.

approche ne gère pas une hausse de la fréquence des données à traiter car les ressources disponibles au sein du SFI définissent une limite physique au temps d'exécution de chaque partie du processus de fusion. La fréquence maximale d'arrivée des données à traiter dépendra donc du système mettant en œuvre le processus de fusion.

Le nombre maximal d'opérations intervenant dans le traitement n'est pour l'heure pas évalué et dépend fortement du SFI utilisé. De plus, la nature des traitements implémentés influencerait cette limite et notre approche doit être indépendante de l'implémentation des processus de fusion déployés.

Positionnement de nos travaux La gestion du passage à l'échelle, positionnée sur la Figure 2.8, est permise par la nature des mécanismes de contrôle. Notre approche propose des solutions aux points suivants :

- Passage à l'échelle du nombre d'éléments contrôlés dans le système,
- Passage à l'échelle du nombre d'éléments utilisés pour contrôler le système.

Les autres dimensions de passage à l'échelle sont hors du cadre de nos travaux.

2.6 Réutilisation et généricité du système

Derrière cet objectif nous regroupons d'une part la capacité à réutiliser tout ou partie des processus de fusion développés et d'autre part la généricité de notre approche par rapport aux modèles de fusion d'informations.

Tout au long de l'étude bibliographique, nous avons été face à une profusion de SFI. Une nouvelle méthode de fusion, un algorithme de traitement plus performant ou des agents plus intelligents sont autant de justifications au développement des systèmes rencontrés et des économies de temps et d'effort sont possibles. Nous avons néanmoins noté que quelques solutions étaient réutilisées, par exemple les systèmes d'exploitation dédiés aux réseaux de capteurs.

La généricité de notre modèle doit permettre la description, le déploiement et le contrôle d'un processus quelque soit le modèle de fusion sous-jacent. Cette indépendance vis-à-vis des modèles théoriques ne s'accompagne pas d'un effort d'adaptation trop important pour un spécialiste en fusion d'informations. Notre modèle de description des processus de fusion est d'une part simple à manipuler et d'autre part rapide à implémenter : un développeur doit fournir l'implémentation du traitement à appliquer sans tenir compte des mécanismes de contrôle. La partie mise en œuvre du mémoire fournit de plus amples détails sur le développement des éléments de fusion.

Une fois le processus de fusion décrit, des économies de temps peuvent être réalisées lors du développement des implémentations des sous-parties du processus. Certaines opérations, comme par exemple, celles liées à l'alignement des données [Wal02], peuvent ainsi être utilisées dans plusieurs applications de fusion d'informations.

De plus, notre approche autorise la connexion d'un autre système de contrôle. En effet, celui que nous proposons résout des objectifs qui peuvent être insuffisants à certains domaines d'applications.

L'administration de notre système peut aussi être totalement modifiée au profit d'une interface contrainte par le domaine applicatif. Ainsi, le système de contrôle que nous proposons peut être entièrement administré par un autre système de plus haut niveau : par exemple pour automatiquement définir le processus de fusion en fonction des traitements disponibles dans le SFI. La Figure 5.1 présente la possible supervision du SFI par un autre système de plus haut niveau.

Positionnement de nos travaux La généricité et la réutilisation du système, positionnée sur la Figure 2.9, est permise par la nature des éléments manipulés. Notre approche propose des solutions aux points suivants :

- Le contrôle de l'exécution du traitement peut être sous-traitée à un autre système *via* un service.
- Le protocole de communication est courant et son extension à d'autres protocoles est possible. Tous les formats de données sérialisables peuvent être déployés.
- L'implémentation du processus de fusion peut se faire dans des bibliothèques natives ou Java. Le développement d'un processus de fusion est réalisé par l'implémentation de traitements élémentaires.
- L'administration du système peut être réalisée par l'interface graphique que nous proposons ou par une autre *via* des services.
- La configuration du système est basée sur un modèle de description du processus de fusion compatible avec les modèles de fusion d'informations les plus courants.

2.7 Synthèse

Hors objectifs

Le système que nous proposons et le modèle théorique sous-jacent, positionnés sur la Figure 2.10, traitent une partie des problématiques identifiées dans les réseaux de capteurs. Ainsi, nous ne souhaitons aborder ni les problématiques liées à la confidentialité des données échangées dans le système ni celles avancées en sûreté de fonctionnement, quelque soit le niveau d'étude.

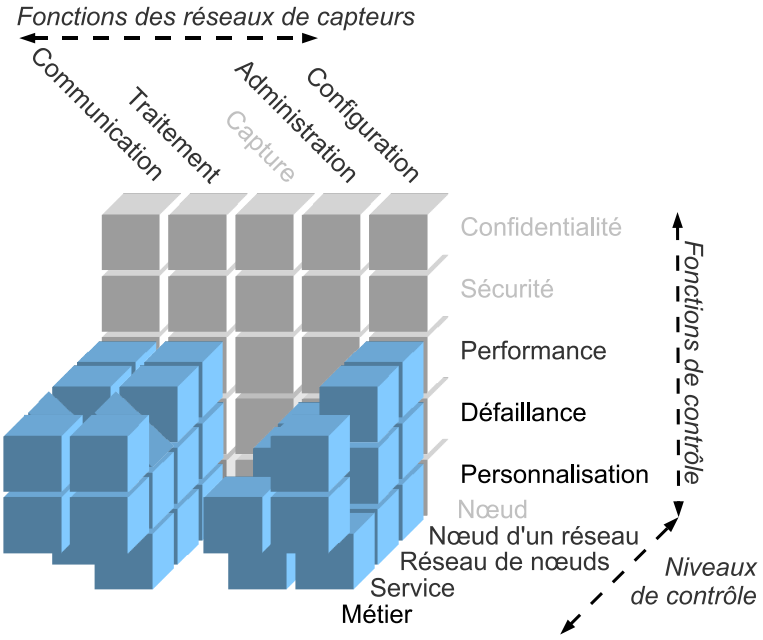


FIG. 2.10 – Contrôle des systèmes répartis de fusion d'informations : positionnement de nos travaux dans le repère traduit de [RNL03].

De même, notre approche ne gère pas la capture du signal produit par un capteur et nous nous situons à un niveau où l'information est déjà définie.

Enfin, cette thèse ne porte pas sur l'étude d'un nœud isolé du système et nos travaux débutent dès la définition d'un réseau de communication.

Concernant nos objectifs, nos travaux proposent des solutions à plusieurs niveaux. La gestion de la volatilité des ressources, Paragraphe 2.3, et l'adaptation de la puissance de calcul allouée au traitement, Paragraphe 2.4, sont atteints par la nature du contrôle que nous définissons. Le Chapitre 5 introduit les tâches de contrôle que nous proposons. Celles-ci gèrent la reconfiguration du système suite à la détection d'une défaillance puis maintiennent et améliorent les performances du système de fusion.

À un autre niveau de conception, la répartition du système, détaillée dans le Chapitre 9, résout notre objectif de passage à l'échelle du système. Ainsi, le système de contrôle que nous proposons exploite l'ajout de nouveaux éléments dans le système tant pour l'exécution du processus de fusion que pour son contrôle. En outre, le mécanisme de répartition que nous proposons apporte une solution auto-adaptative de l'architecture du système de contrôle.

Notre modèle de déploiement favorise quant à lui la réutilisation des éléments d'un processus de fusion principalement par la généricité de notre modèle de description. De plus, la granularité fine de l'implémentation du processus de fusion permet la réutilisation de codes exécutables entre plusieurs processus.

Première partie

État de l'art

Chapitre 3

Étude des systèmes répartis de fusion d'informations

La fusion d'informations est un domaine de recherche dont l'étude bibliographique est difficile si l'on souhaite dresser une vue instantanée et approfondie des productions scientifiques de la communauté. Le caractère réparti des SFI élargit encore la quantité de travaux à étudier. Nous avons pour autant réussi à dégager des axes d'étude que nous présentons dans ce chapitre. Nous avons choisi de formuler deux niveaux d'étude des SRFI vis-à-vis desquels nous positionnons notre contribution : le matériel et les outils logiciels.

Les Figures 3.1 et 3.2 présentent le fil conducteur de ce chapitre. L'axe horizontal représente le niveau d'abstraction de l'étude des SRFI et il s'accompagne de l'idée qui a guidé notre analyse : un exemple de problème posé et le type de solutions apportées sont à chaque fois introduits.

Des publications étudiées durant la thèse, celles qui traitent de la mise en œuvre d'un processus de fusion sont à classer dans trois catégories. La première regroupe les applications d'un processus de fusion dans la résolution d'un problème, comme par exemple la navigation autonome d'un robot. Peu de détails sont alors donnés sur le système sous-jacent, l'accent étant alors mis sur la méthode de fusion employée. Les détails d'un système de fusion, d'un outil de mis en œuvre, sont généralement fournis dans les publications de la seconde catégorie. Ces même outils sont alors étudiés dans la troisième catégorie d'articles, dont le classification « support d'exécution, système d'exploitation, abstraction, cadre de fusion » émerge des panoramas dressés. Nous brossons dans ce mémoire les différents types de solutions exploitées dans la mise en œuvre d'un système de fusion d'informations et nous approfondissons le point consacré aux outils logiciels existants.

3.1 Le matériel de mise en œuvre

Au plus proche du monde observé, le niveau consacré au matériel propose des solutions d'exécution des processus de fusion. Ce domaine de recherche répond à des problèmes comme la fiabilité des communications ou le stockage des informations en attente pour ne citer qu'une partie des objectifs énoncés dans [IM04]. Les solutions qui composent ce niveau répondent aussi à des problèmes liés à la conception des capteurs et des éléments de traitement. Concernant les capteurs, la communauté propose des améliorations de leur précision ou leur miniaturisation. D'autres améliorations concernent la consommation énergétique des composants, leur résistance à des conditions extrêmes comme le vide et les hautes températures ou encore la puissance de traitement [CES04]. La recherche agronomique déploie par exemple ce genre de capteurs dans le sol des exploitations agricoles pour mesurer la composition de la terre avant l'épandage d'engrais

FIG. 3.1 – Premier niveau d'étude des systèmes répartis de fusion d'informations : le matériel de mise en œuvre.

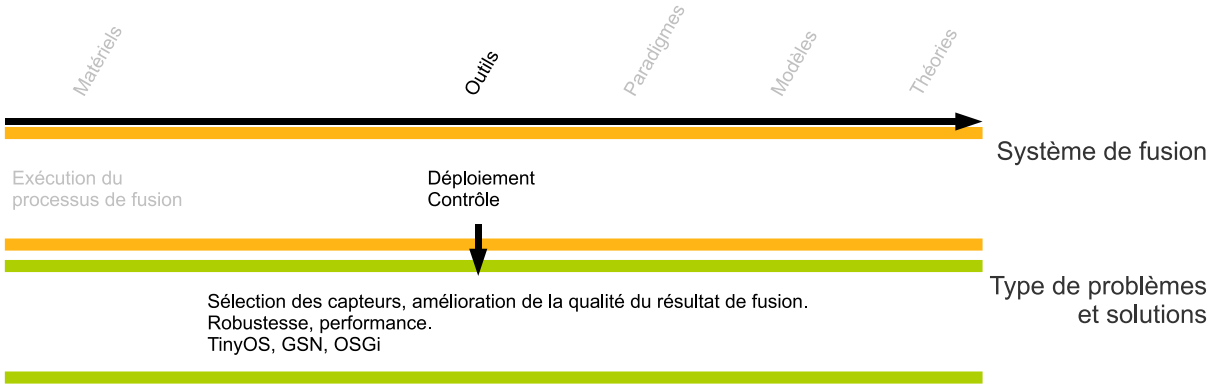


FIG. 3.2 – Deuxième niveau d'étude des systèmes répartis de fusion d'informations : les outils de mise en œuvre.

pour en adapter la concentration.

Les solutions proposées à ce niveau conceptuel sont de nature hétérogène et il existe des SRFI mis en œuvre grâce à des solutions dédiées aux réseaux de capteurs, comme par exemple les motes ou les nœuds de capteurs, ou par des architectures comme des PC et des assistants électroniques.

Ce niveau d'étude des systèmes de fusion d'informations sort du cadre de nos travaux et nous limitons le cadre applicatif de nos travaux aux architectures présentées dans le Chapitre 2.2.2.

3.2 Les outils de mise en œuvre

L'étude des SRFI peut également avoir lieu au niveau des outils logiciels permettant la mise en œuvre des processus de fusion. Ce niveau peut lui même être divisé en niveaux intermédiaires selon le niveau d'abstraction des solutions proposées. La plupart de celles-ci sont indépendantes de la méthode de fusion d'informations implémentée et de la couche matérielle sous-jacente. La nature de ces systèmes est très hétérogène d'un niveau de mise en œuvre à l'autre. Citons par exemple TinyOS [LMG+04], qui illustre l'étude des systèmes d'exploitation dédiés aux réseaux de capteurs et GSN [Sal07] présenté au niveau des plates-formes de service de fusion d'informations.

Niveau d'abstraction	Nom du système
Systèmes d'exploitation	TinyOS, Mantis, SOS Contiki, t-kernel
Cadres d'exécution	Maté, Melete, DVM DAViM, VM*, SwissQM OSM, DSN, FACTS
Plates-formes de fusion d'informations	UPnP, Jadabs, Chedar Mobile Chedar, Proem Active Sensor Network (ASN) Sensor bean
Plates-formes services	MidFusion, COMiS, TinyLIME, Agimone Janus, SwissQM Gateway SONGS, JDDAC, GSN

TAB. 3.1 – Niveaux intermédiaires de l'étude des outils de mise en œuvre des systèmes répartis de fusion d'informations.

Ce paragraphe présente les quatre niveaux intermédiaires inspirés de [HMM+07] et illustre chacun d'eux par une sélection d'outils disponibles. Notre sélection est basée sur les caractéristiques que nous souhaitons reprendre ou améliorer dans nos travaux. Un paragraphe spécial en fait la synthèse à la fin de ce chapitre.

3.2.1 Les systèmes d'exploitation dédiés

Le niveau d'abstraction le plus bas est composé des systèmes d'exploitation directement connectés au matériel du SFI. TinyOS [LMG+04], Mantis [BCD+05] et Sensor Operating System (SOS) [HKS+03] sont trois des systèmes d'exploitation incontournables des réseaux de capteurs. Ce type de systèmes apporte des solutions aux problèmes de gestion de la mémoire et de l'ordonnancement des tâches à exécuter. Les fonctionnalités les plus répandues, organisées en modules activables indépendamment, gèrent les communications et dans certains cas la découverte du voisinage réseau.

Un catalogue des systèmes d'exploitation dédiés aux réseaux de capteurs [HMM+07] est ici inutile car notre approche ne se situe pas au même niveau que ce type de contributions.

3.2.2 Les cadres d'exécution dédiés

Nous regroupons ici les abstractions des routines du système d'exploitation, comme les bibliothèques d'instructions ou les machines virtuelles. Ces solutions, ici dédiées aux réseaux de capteurs, proposent par exemple la migration de code pendant l'exécution du processus ou des mécanismes de reconfiguration du système.

Maté [LC02] est un de ces cadres d'exécution. Basé sur TinyOS, Maté permet la construction d'applications exécutées dans des machines virtuelles dans le but de réutiliser des blocs d'applications. Melete [YRBL06] étend Maté par la gestion d'applications concurrentes au sein d'une même machine virtuelle. D'autres exemples de cadres d'exécution, comme DVM [BHK+06], sont disponibles pour d'autres systèmes d'exploitation que TinyOS.

Ici encore, les solutions apportées par ces systèmes ne se situent pas au même niveau que notre approche. *Primo*, les systèmes de ce niveaux sont basés sur des systèmes d'exploitation dédiés aux réseaux de capteurs qui sont à la frontière basse des architectures que nous visons. *Secundo*, les problèmes traités par ces systèmes sont également différents des nôtres. La concurrence de

deux applications ou la gestion du niveau d'énergie disponible sortent du périmètre de notre étude.

3.2.3 Les plates-formes de fusion d'informations

Ce groupe de solutions de mise en œuvre est composé des plates-formes de fusion basées sur des systèmes d'exploitation conventionnels (Microsoft Windows, Linux ou MacOS). Le type d'architectures de ces plates-formes ne se limite donc pas aux réseaux de capteurs. Outre cette caractéristique, les plates-formes de ce niveau sont utilisées pour résoudre un problème de fusion fixé et sont liées au domaine d'application visé. De plus, ces plates-formes sont « fermées » car elles intègrent tous les éléments physiques du système, depuis les sources jusqu'aux actionneurs. La réutilisation des éléments développés reste difficile et le résultat du processus de fusion ne sort pas du système.

Ainsi, Active Sensor Network (ASN) [MD06] est une plate-forme de fusion pour la collecte autonome et coopérative d'informations, supportant le passage à l'échelle. Celui-ci se justifie par le déploiement du système sur un nombre pouvant être important de plates-formes, avec comme objectif la garantie d'un niveau de performance. Le redéploiement du système est réalisé de façon autonome et automatique, dans le but de former un réseau capable de s'adapter aux modifications de connexion. Même si les plates-formes peuvent fonctionner seules, elles œuvrent ensemble pour atteindre un objectif commun si leur ressources le leur permettent. ASN est divisé en deux ensembles, un composé de plates-formes robotiques équipées de capteurs ainsi que d'actionneurs et un composé d'opérateurs humains. Les plates-formes robotiques communiquent ensemble et forment le réseau de capteurs avec lequel les opérateurs interagissent *via* des interfaces dédiées. L'intérêt de ASN réside dans sa capacité à modifier sa configuration de façon autonome. De plus, ASN, ou plus généralement les plates-formes de fusion, est générique et indépendant du modèle de fusion choisi (par exemple ASN met en œuvre un processus de fusion bayésien dans [MBW+04] mais d'autres modèles sont compatibles).

Sensor bean [MD05] est également une plate-forme de fusion, dans laquelle la gestion de la dynamicité des connexions repose sur le typage des ports de communication. Cette approche, originale, définit trois type de ports : pour les requêtes synchrones, pour les requêtes asynchrones comme les notifications d'événements et pour les données typées. Cette sémantique des ports place le problème de la sélection des services ou des capteurs au niveau de la conception d'un Sensor Bean.

Dans un souci de clarté, nous présentons dans le Paragraphe 3.3 une synthèse et le positionnement de notre outil logiciel par rapport aux systèmes précédents.

3.2.4 Les plates-formes de services de fusion d'informations

Les systèmes qui composent ce groupe, en plus de résoudre les même problèmes que ceux du groupe précédent, mettent à la disposition des tiers logiciels des mécanismes de contrôle : le plus simple étant la sortie des résultats issus du processus de fusion. Par ailleurs, certains des systèmes de ce niveau proposent le déploiement d'autres types de processus.

Par exemple, MidFusion [HKS04] est un *middleware* entre un ensemble de capteurs et une application. Le but de MidFusion est la découverte et la sélection du meilleur sous-ensemble de capteurs répondant à un problème tout en respectant une qualité de service. Cette sélection a lieu avant l'exécution du processus et le problème résolu par MidFusion peut être énoncé comme ceci : *« Soit un ensemble de capteurs producteurs de données et la qualité de service requise par une application, quel est le meilleur sous-ensemble de capteurs que le middleware peut utiliser pour maximiser la fusion d'informations tout en minimisant les coûts ? »* . Ce mécanisme de sélection suppose que les coûts d'utilisation d'une ressource sont constants. Il est donc clair que

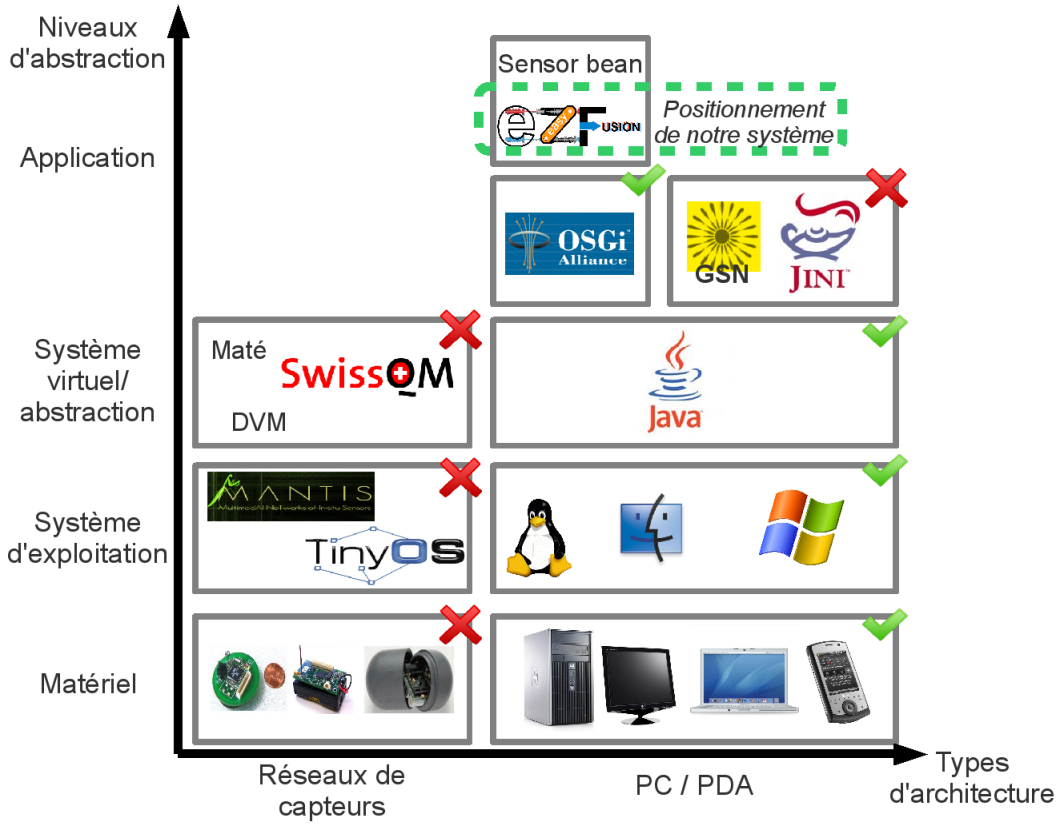


FIG. 3.3 – Positionnement de nos travaux par rapport aux solutions existantes.

MidFusion est développé dans le but d'être appliqué à un ensemble de capteurs statiques et dans un environnement maîtrisé, tout du moins dans leurs caractéristiques. Outre cette évaluation du coût d'une configuration, MidFusion applique une méthode de fusion d'informations basée sur des réseaux bayésiens uniquement.

COMiS [JVN05] (abréviation de *Component Oriented Middleware for Sensor networks*) est un autre exemple de système de ce groupe, indépendant du modèle de fusion choisi. COMiS est un *middleware* entre une ou plusieurs applications et un ensemble de capteurs regroupés en nœuds. L'ensemble des nœuds est statique et seul le *middleware* peut interroger les nœuds. COMiS est écrit en DCL (*Distributed Compositional Language*) et se divise en six types de composants. Les *listener* reçoivent toutes les requêtes et les transmettent au voisinage ou les ajoutent à une file d'attente locale. Les *register* enregistrent les composants locaux afin que les *discovery* localisent les composants au sein du réseau. Les *sender* envoient les données entre les nœuds présents dans le réseau. Les *updater* déploient et mettent à jour les composants. Les *power manager* font le lien entre les routines de gestion d'énergie de l'OS. Les tâches de contrôle affectées aux composants de COMiS ont été développées selon une approche « système réparti ». Le contrôle réalisé des COMiS est donc formé d'un regroupement de composants alors que d'autres approches proposent un mécanisme global qui s'exécute sur un ensemble de composants.

3.3 Synthèse

L'étude des SFI existants a été la première étape des travaux de thèse qui ont abouti à ce mémoire. Le large spectre des solutions proposées, ou utilisées, dans les publications recueillies pendant l'élaboration de l'état de l'art nous a permis de préciser quels étaient les problématiques que nous avons résolues. Le Chapitre 2.2.2 détaille ces problématiques et nous discutons ici du positionnement de notre système par rapport à ceux déjà disponibles - la Figure 3.3 illustre ce positionnement. Une étude plus précise de GSN [Sal07], a donné lieu à la publication [PBHM08, BHMP07] de nos travaux sur un *wrapper* entre le protocole de contrôle OPC [Shi99] et le système GSN.

Le premier de nos choix concerne le niveau d'exécution de notre système par rapport au matériel que nous souhaitons gérer. Il est rapidement apparu que nous ne nous destinions pas à la création d'un nouveau système d'exploitation dédié aux réseaux de capteurs. De plus, les problèmes de gestion d'énergie n'entraient pas dans le cadre d'un contrôle tel que nous le présentons dans le Chapitre 5.

Outre le choix du développement d'un système de plus haut niveau, nous avons également dû choisir le système d'exploitation sur lequel nous nous basons. Ainsi, trois perspectives s'offraient à nous selon le public que nous visons et la flexibilité que nous souhaitons donner à notre système : un système d'exécution dédié aux réseaux de capteurs, un système d'exploitation pour PC ou un système d'exploitation virtuel. Les systèmes liés aux réseaux de capteurs ont été écartés au profit des systèmes « grand public » accessibles aussi bien par les laboratoires de recherche que par l'utilisateur final. De plus, nous avons décidé de fonder notre système sur des machines virtuelles Java, principalement car seule cette solution nous permet de gérer simplement la reconfiguration du système à chaud, tout en étant disponible sur la plupart des systèmes d'exploitation rencontrés.

De plus, nous préférons concentrer nos efforts sur la reconfiguration automatique du système et sur le caractère autonome du système de contrôle. Certaines problématiques de gestion « fine » des ressources disponibles sortent du périmètre de notre étude. Nous ne cherchons donc pas à optimiser l'espace mémoire utilisé ou même l'ordonnancement de processus concurrents.

Nous avons enfin choisi d'ouvrir le champ des modèles de fusion compatibles à ceux les plus utilisés. Le chapitre suivant présente les différents paradigmes et modèles de fusion possibles et nous expliquons ceux compatibles dans le Chapitre 6.

Chapitre 4

Étude des modèles de fusion

Ce second chapitre de l'état de l'art présente la fusion d'informations selon une vue plus abstraite que dans le Chapitre 3. Trois paragraphes présentent les niveaux d'abstraction que nous avons identifiés. Les paradigmes de mise en œuvre proposent un cadre de développement indépendant du modèle de fusion choisi et du système logiciel sous-jacent. Les deux autres paragraphes proposent respectivement un panorama des modèles de fusion et une présentation de la théorie de fusion d'informations.

Les Figures 4.1, 4.3 et 4.4 reprennent la notation rencontrée dans le Chapitre 3. L'axe horizontal représente le niveau d'abstraction de l'étude des SRFI et il s'accompagne de l'idée qui a guidé notre analyse

4.1 Les paradigmes de mise en œuvre

Nous définissons par paradigme une méthode de mise en œuvre indépendante de toute solution technique et compatible avec plusieurs modèles de fusion d'informations. Pour illustrer ce niveau d'étude des SFI, nous avons sélectionné trois paradigmes souvent rencontrés dans l'étude bibliographique. Le premier, les réseaux neuronaux, est souvent utilisé pour la mise en œuvre d'un processus d'identification. Le second est basé sur le concept de producteur-consommateur, c'est également le plus simple de ceux rencontrés. Enfin, le dernier paradigme que nous avons identifié, propose l'utilisation de Systèmes Multi-Agents (SMA).

4.1.1 Réseaux neuronaux

Nombre de processus de fusion d'informations sont basés sur les réseaux neuronaux [BBH07]. Un réseau de neurones est composé d'éléments simples inspirés par le système nerveux biologique. Les neurones fonctionnent en parallèle et chacun réalise une fonction $f(y)$ d'une somme pondérée y de signaux $x_0, x_1, \ldots, x_n$ qui lui parviennent. Les coefficients de pondération de la somme $y = \sum_{i=0}^{n} w_i x_i$ s'appellent les poids synaptiques et s'appliquent sur les connexions d'un neurone prises sur les sorties de la couche précédente. Si w_i est positif, l'entrée x_i est excitatrice alors que si w_i est négatif, elle est inhibée.

L'apprentissage des réseaux de neurones peut être supervisé ou non. Dans le premier type, une entrée particulière, ou cible, est présentée en entrée du réseau et les poids synaptiques sont ajustés pour que le résultat produit en sortie du réseau corresponde à la cible. Dans le cas d'un apprentissage non supervisé, des exemples ou « patrons » sont présentés au réseau qu'on laisse s'auto-organiser au moyen de lois locales qui régissent l'évolution des poids synaptiques.

La fonction d'activation $f(y)$(ou fonction de seuil) sert à introduire une non-linéarité dans le fonctionnement du neurone. Cette fonction présente généralement trois intervalles : en dessous

FIG. 4.1 – Troisième niveau d'étude des systèmes répartis de fusion d'informations : les paradigmes de mise en œuvre.

du seuil et le neurone est inactif, aux alentours du seuil et le neurone est en phase de transition, enfin au-dessus du seuil et le neurone est actif. C'est au sein de cette fonction d'activation que les modèles de fusion probabiliste ou floue peuvent intervenir. Les réseaux neuronaux sont le plus souvent utilisés pour la mise en œuvre d'un modèle de fusion, comme par exemple [CGM+92] qui déploie un processus de fusion floue et [Wil96] où les neurones implémentent des fonctions de fusion gaussiennes.

4.1.2 Producteur-consommateur

Ce paradigme de mise en œuvre de SFI, illustré par la Figure 4.2, décrit le traitement séquentiel des informations. Il propose de découper un processus en sous-processus concurrents, connectés dans un réseau de files d'attente. Le but de ce modèle est de pouvoir gérer le partage d'informations partiellement traitées entre les opérations d'un processus de fusion.

Ce modèle est celui le plus souvent rencontré : la plupart des plates-formes de fusion et des plates-formes de services que nous avons rencontrées sont également basées sur ce paradigme (c.f. Paragraphe 3.2). De plus, la description de la composition d'informations à partir de plusieurs sources est explicitée dans la description du SFI.

La difficulté de mise en œuvre du modèle producteur-consommateur réside dans la synchronisation du dépôt d'une information dans une file d'attente par le producteur, et la lecture de cette information par le consommateur. Excepté cette difficulté, facilement contournée par l'utilisation des outils de conception et de développement disponibles, ce modèle est le plus simple à mettre en œuvre.

4.1.3 Systèmes multi-agents

Au même niveau que les réseaux neuronaux, nous avons choisi de présenter les SMA comme un paradigme de mise en œuvre des SFI. D'une manière générale, les SMA sont utilisés dans les cas suivants [Flo99] : (1) lorsque chaque agent peut participer à la résolution d'un problème, (2) quand aucun système de contrôle central n'est disponible, (3) si les données nécessaires à la résolution du problème ne sont pas localisées dans un même lieu et (4) que les calculs mis en œuvre pour la résolution du problème font intervenir des communications. Dans un SMA, les agents doivent coopérer de telle sorte que c'est le groupe d'agents qui va résoudre le problème posé car un agent seul ne peut le faire. Une des difficultés rencontrées pendant la création de tels systèmes réside dans la composition variable desdits groupes. Des mécanismes statiques liés aux

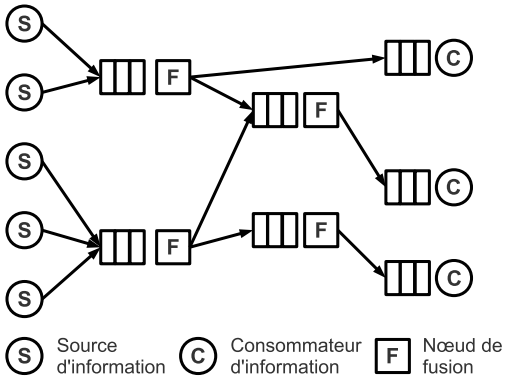


FIG. 4.2 – Modèle de conception producteur-consommateur : les flèches représentent le sens des échanges d'informations.

domaines d'application sont minoritaires car la généricité du système et sa réutilisation sont les principaux objectifs des SMA.

Les liens tissés entre les SMA et les SRFI peuvent revêtir trois aspects. Comme nous l'avons noté dans l'étude des systèmes de fusion d'informations basés sur les SMA, la première utilisation possible réside dans la fusion proprement dite. Les agents d'une communauté échangent leurs analyses d'une situation et procèdent par composition, ou comparaison, des connaissances de chacun pour élaborer un résultat final.

Ensuite, viennent les systèmes dont le contrôle repose sur un SMA. Les agents basent par exemple le contrôle du système sur des protocoles de négociation. Ce type de système propose donc une dichotomie des rôles avec d'une part le processus de fusion d'informations, et d'autre part le contrôle du système.

Enfin le dernier domaine commun aux SMA et aux SRFI concerne l'exploitation d'un processus de fusion, au sein d'un agent, pour que celui-ci soit en mesure de prendre des décisions. L'utilisation des SMA n'a dans ce dernier cas aucun lien avec la fusion d'informations qui est alors un outil au service des SMA.

La suite de ce paragraphe présente une sélection d'applications des SMA pour la fusion d'informations. La limite entre les trois cas que nous venons de lister est dans certains cas ténue, ce qui démontre une certaine ambiguïté dans le positionnement des auteurs.

Lors de l'étude bibliographique, nous avons trouvé nombre d'exemples d'applications basés sur les structures et les protocoles admis dans la communauté. C'est le cas de [APPJ08] où les auteurs proposent d'implémenter un Système de Gestion Domotique (SGD) par un SMA. Un SGD se caractérise par la répartition des agents, guidée par leur position dans l'habitat ou par les emplacements des sources d'énergie. L'évolution de la disponibilité des ressources, tant par leur nombre que par leurs attributs, caractérise également un SGD. Le contexte ouvert définit la dernière caractéristique d'un SGD. Ainsi, les agents doivent être en mesure d'intégrer de nouveaux éléments sans qu'il soit nécessaire de redéfinir les mécanismes de contrôle. L'originalité de l'approche présentée dans [APPJ08] réside dans l'interaction de deux mécanismes de contrôle concurrents. Le premier mécanisme gère les urgences déclenchées au sein du système lorsqu'un agent est en manque d'énergie, ou plus généralement de ressources. Le second mécanisme anticipe l'évolution du système afin d'éviter les erreurs possibles.

[CCL07] propose un autre exemple d'utilisation d'un SMA basé sur le modèle d'agents mobiles [Whi96]. Un tel agent est exécuté sur des nœuds physiques et peut migrer grâce à la transmission d'un message qui décrit le comportement de l'agent ainsi que son état d'exécution. Cette

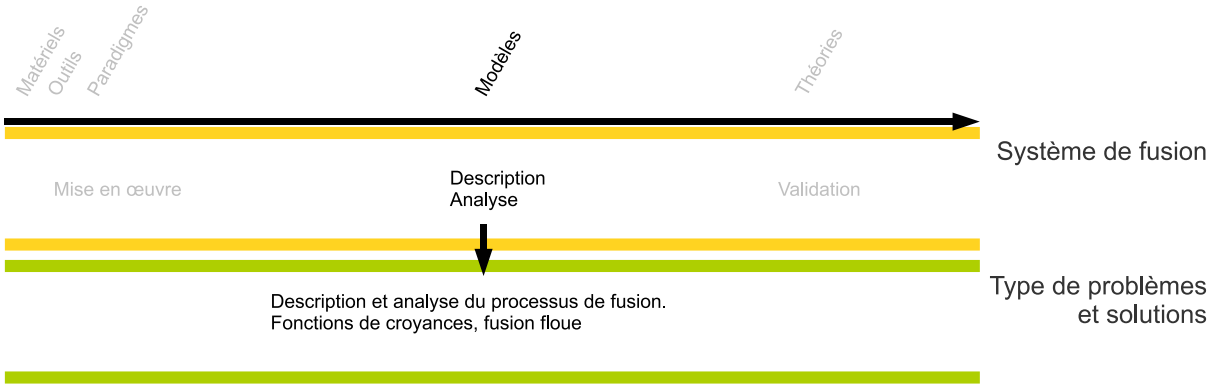


FIG. 4.3 – Quatrième niveau d'étude des systèmes répartis de fusion d'informations : les modèles de fusion.

approche s'est heurtée aux problèmes de standardisation et de sécurité. Il est en effet difficile de réutiliser une partie du système avec une autre architecture, même si les nœuds qui le composent exposent leur méthodes *via* des services Web.

Les applications des SMA pour la fusion d'informations sont nombreuses et nous concluons ce panorama non exhaustif des solutions disponibles par un exemple de SMA mettant en œuvre un SRFI qui illustre parfaitement la confusion souvent faite entre des agents utilisés pour la fusion d'informations et pour le contrôle du système. En effet, l'approche présentée dans [PON05] propose dans un premier temps le recensement des sources d'informations disponibles au moment de l'exécution du processus de fusion. Par la suite, un sous-ensemble de ces sources d'informations est sélectionné puis lié à un modèle probabiliste du monde. La fusion d'informations s'effectue hiérarchiquement en deux étapes : la spécialisation d'une requête, énoncée par l'expert, pendant la descente vers les sources d'informations et l'agrégation des résultats pendant la remontée dans le réseau. En parallèle de ce mécanisme de fusion hiérarchique, le système proposé exécute une gestion décentralisée des sources d'informations présentes dans le système.

4.2 Les modèles de fusion

Ce paragraphe présente les modèles de fusion d'informations rencontrés lors de l'étude des SFI existants. [Blo05] propose un panorama utile aux non-initiés et [BDPP06] détaille les modèles que nous présentons.

Fusion d'informations Les modèles et méthodes de fusion d'informations sont utilisés lors de la conception du traitement appliqué aux informations. Les modèles fournissent le cadre théorique nécessaire à la description des informations consommées et produites en résultat, ainsi que la méthode à appliquer pour mettre en œuvre le processus de fusion. Par ailleurs, indépendamment de la méthode de fusion utilisée, il est également possible de classer un processus de fusion selon le niveau hiérarchique du traitement appliqué.

[SPU99] présente par exemple une fusion d'informations selon trois niveaux : le pixel, le voisinage de pixels et lors de la prise de décision. L'identification s'effectue à chaque niveau et pour chaque zone traitée, un degré d'appartenance est calculé pour chacune des classes de l'espace de discernement.

D'autres hiérarchies sont présentées dans [BP02, SBW98] et l'étude de ce type de classification sort du cadre de nos travaux.

Plusieurs catégorisations des processus de fusion d'informations existent, par exemple celle proposée dans [Sas02] : les modèles probabilistes, les méthodes d'approximation et les approches « intelligentes ». Les processus de fusion du premier groupe utilisent le plus souvent des approches statistiques ou bayésiennes, le second groupe de processus de fusion regroupe les méthodes d'approximation du système observé comme par exemple des filtres de Kalman ou des méthodes du moindre carré. Les attributs du modèle du système observé sont mis à jour pendant l'exécution afin de prédire le prochain état du système. Les approches restantes sont qualifiées d'« intelligentes » et regroupent les modèles flous, les réseaux neuronaux [PC03] et les algorithmes génétiques [Mit98]. Bien entendu ces groupes ne sont pas exclusifs et il est possible d'utiliser des approches mixtes lors de la conception de systèmes de fusion d'informations. Nous avons noté l'absence des SMA dans cette classification. Celle-ci s'explique par l'utilisation, le plus souvent, des SMA comme d'un outil de mise en œuvre des processus de fusion. Ceux-ci ont plus généralement la charge de la sélection des sources d'informations pertinentes [ZAR05], des choix des traitements à appliquer ou de la répartition du processus de fusion [BQX08].

La suite de ce paragraphe présente les différents modèles de fusion d'informations rencontrés pendant l'étude des SFI existants : la fusion probabiliste non bayésienne, la fusion probabiliste bayésienne, la fusion en théorie des possibilités et la fusion de fonctions de croyances.

Nos travaux se concentrent sur les traitements des informations comme la combinaison d'informations et la prise de décision. Les analyses basées sur le résultat du processus de fusion et la représentation de l'information [Blo05] sortent du cadre de notre étude.

4.2.1 Fusion probabiliste

Non bayésienne Dans ce modèle, on suppose que la source S_i fournit une mesure de probabilité P_i sur chaque élément de l'ensemble de discernement $d_i \in D$. La seule fonction de fusion possible [LW81, McC81] est la moyenne pondérée des données consommées telle que la somme des poids, positifs ou nuls, fasse 1. Le résultat de cette fusion est la probabilité qu'un objet observé appartienne à la classe $d \in D$.

Bayésienne Les informations manipulées dans le cadre de la théorie probabiliste s'écrivent sous la forme $p(d_i|S_j)$ où S_j est la source et d_i un élément de l'ensemble de discernement. En général, la valeur de p est estimée et on en déduit la probabilité par application de la règle de Bayes. Dès lors que les informations peuvent être modélisées sous la forme de probabilités, la fusion (ou combinaison [Blo05]) d'informations peut être effectuée de la manière suivante :

$$p(d_i|S_1, \ldots, S_l) = \frac{p(S_1, \ldots, S_l|d_i)p(d_i)}{p(S_1, \ldots, S_l)}$$

Cette fusion, difficile à effectuer si les statistiques sont insuffisantes, peut être simplifiée sous l'hypothèse d'indépendance (le calcul de critères de validité de cette hypothèse sort du cadre de notre étude). La formule précédente devient alors un produit des probabilités et la fusion des informations nécessite alors l'estimation *a priori* des probabilités des décisions $p(d_i)$.

Décision À la suite de la fusion probabiliste des informations disponibles, la prise de décision est basée sur le calcul d'un critère *a posteriori*. La règle la plus utilisée est celle de la valeur maximale qui sélectionne la classe $d \in D$ la plus probable.

4.2.2 Fusion possibiliste

La théorie des possibilités permet de modéliser des connaissances imprécises ou vague, et autorise donc le traitement conjoint de connaissances fournies numériquement et de connaissances

exprimées symboliquement, notamment par l'utilisation de sous-ensembles flous [ZKY96]. Elle permet de manipuler dans un même formalisme des imprécisions et des incertitudes comme peut le faire une fonction de croyances.

La mesure de possibilité est définie comme une fonction qui pour tout sous-ensemble d'un ensemble de référence X associe un coefficient compris entre 0 et 1, ce qui se traduit par la possibilité d'un événement $x \subseteq X$. Cette mesure de possibilité Π vérifie les points suivants :

- $\Pi(\emptyset) = 0$, $\Pi(X) = 1$;
- $\forall A_1 \subseteq X$, $A_2 \subseteq X ... \ \Pi(\cup_{i=1,2...} A_i) = sup_{i=1,2...} \Pi(A_i)$.

Alors que la mesure de possibilité s'applique à un sous-ensemble d'un ensemble de référence, la distribution d'une possibilité définit le coefficient attribué aux éléments de X. Un sous-ensemble est construit par l'union des éléments de X et un coefficient de possibilité peut être affecté en se servant du point précédent. On appelle distribution de possibilité π une fonction définie sur X à valeurs dans $[0, 1]$ qui satisfait la condition suivante : $sup_{x \in X} \pi(x) = 1$.

Concernant la fusion d'informations, chaque source S_i fournit une distribution de possibilité π_i sur l'ensemble de discernement D dans lequel doit être classé un objet observé. Les opérateurs d'agrégation possibilistes sont alors appliqués aux distributions provenant des sources. Le résultat de cette fusion est généralement aussi une distribution de possibilités.

La notion de possibilité a été introduite pour quantifier en partie la certitude d'une classe, ou d'un sous-ensemble, ordinaire d'un ensemble de référence. Lorsque les classes étudiées sont floues, on peut indiquer dans quelle mesure elles sont possibles, étant données des connaissances préalables sur l'ensemble de référence.

La construction des fonctions d'appartenance peut être effectuée par apprentissage probabiliste ou par heuristique pour ne citer que deux exemples. [Bez81] dresse un panorama des méthodes de construction des fonctions d'appartenance. Les deux étapes du processus de fusion, proposées dans [Blo05], traitent de la combinaison de données floues et de la prise de décision suite à cette combinaison. Une des caractéristiques des opérateurs mettant en œuvre la combinaison est qu'ils fournissent un résultat de même nature que les données consommées. Dans la théorie des ensembles flous, de multiples combinaisons sont possibles, par exemple les t-normes, les t-conormes [Men42, SS05], les moyennes dont les détails sortent du cadre de nos travaux.

Décision À l'issue de la combinaison de données floues, le processus de fusion décrit une étape de décision afin de catégoriser l'objet ou l'événement considéré. La règle principalement utilisée en fusion floue est le maximum des degrés d'appartenance [Blo05] :

$$d_i \text{ si } \mu_i(x) = max\{\mu_k(x), 1 \leq k \leq n\}$$

où μ_k désigne la fonction d'appartenance à la classe k résultant de la combinaison.

4.2.3 Fusion de fonctions de croyances

Les fonctions de croyances permettent de représenter l'imprécision et l'incertitude d'une information. La fonction de plausibilité Pls, la fonction de croyance Bel et la fonction de masse m sont à la base de la représentation de l'information [GB92]. Contrairement à la théorie probabiliste où la probabilité qu'une occurrence d'événement appartienne à une unique classe, la fonction de masse, de par sa définition, autorise un raisonnement sur des sous-ensembles de l'ensemble des décisions possibles.

Dans ce modèle $D = \{d_1, \ldots, d_n\}$ représente l'espace de discernement où chaque d_i désigne une hypothèse en faveur de laquelle une décision peut être prise.

Une fonction de masse m est définie comme une fonction de 2^D dans $[0, 1]$ telle que

$$\sum_{A \subset D} m(A) = 1, \text{ et } m(\emptyset) = 0$$

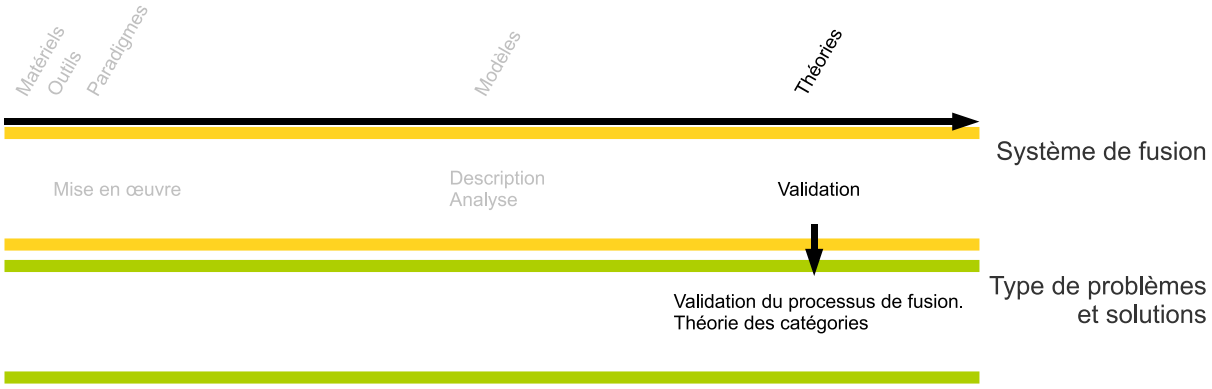


FIG. 4.4 – Cinquième niveau d'étude des systèmes répartis de fusion d'informations : la théorie de la fusion.

Une fonction de croyance Bel est une fonction totalement croissante définie de 2^D dans $[0,1]$ telle que

$$\forall A_1, \ldots, A_k, Bel(\cup_{i=1\ldots k} A_i) \geq \sum_{I \subset \{1\ldots k\}, I \neq \emptyset} (-1)^{|I|+1} Bel(\cap_i \in I A_i)$$

où $|I|$ est le cardinal de I, et tel que $Bel(\emptyset) = 0$ et $Bel(D) = 1$.

De la fonction de croyance découle une formulation de la fonction de masse

$$\forall A \in 2^D, m(A) = \sum_{B \subset A} (-1)^{|A-B|} Bel(B)$$

La fonction de plausibilité Pls, de 2^D dans $[0,1]$, est alors formulée

$$\forall A \in 2^D, Pls(A) = \sum_{B \cap A \neq \emptyset} m(B) = 1 - Bel(A^C)$$

La fusion des informations intervient lors de la combinaison des fonctions de masses effectuée selon la règle orthogonale de Dempster [Sha76].

4.3 La théorie de la fusion d'informations

La fusion d'informations peut être étudiée comme une discipline indépendante des applications et des modèles de fusion sous-jacents. Des modèles du système de fusion permettent alors la comparaison de méthodes de fusion d'informations. De tels modèles sont à destination du concepteur du processus de fusion d'informations qui peut ainsi choisir la méthode de fusion adaptée à l'application envisagée.

Ainsi, [KTW04] définit un modèle de comparaison qui propose la formulation de théorèmes sur le monde observé pour prendre en compte les dépendances et les contraintes auxquelles le monde obéit. Dès lors que le processus de fusion est décrit, celui-ci est classé dans une catégorie servant de base à la comparaison des méthodes de fusion. Le modèle formel de fusion d'informations proposé dans [KTW99] est basé sur la théorie des catégories. La description du processus de fusion vient de quatre outils présentés dans ce modèle. « Le monde » sert d'espace de définition aux objets observés et « le processus de fusion » permet la composition de modèles liés aux données. En effet, des « classes de modèles » représentent les données fournies par les capteurs, les opérations

sur les données et les relations entre les données. Les « théories », ou spécifications, regroupent les connaissances sur le monde, les connaissances sur les capteurs et les buts (les requêtes) sur le monde et complètent les outils du modèle. Le principe de base de [KTW99] est de construire dans un premier temps une théorie fusionnée puis de trouver une classe de modèles associés à cette théorie. Si une telle classe existe, alors le processus de fusion est validé.

Ce niveau d'étude de la fusion d'informations sort du cadre de nos travaux. Par ailleurs, les publications rencontrées dans l'étude bibliographique qui traitent de ce niveau d'étude sont marginales.

4.4 Synthèse

Ce chapitre montre le large spectre des niveaux d'études de la fusion d'informations. Trois niveaux, que nous pouvons résumer par l'organisation d'un processus de fusion, la nature du traitement et l'analyse d'un processus, nous guident dans les choix que nous faisons dans la suite de ce mémoire. En effet, parmi les paradigmes de mise en œuvre d'un processus de fusion, nous avons sélectionné le paradigme producteur-consommateur car il permet la description d'un processus de fusion séquentiel. Ce type d'organisation du processus est compatible avec la plupart des modèles de fusion que nous avons rencontrés, notamment ceux que nous avons sélectionnés dans ce chapitre. L'ouverture de nos travaux au niveau d'étude le plus haut, introduirait de nouveaux problèmes que nous ne souhaitons pas traiter, comme par exemple la notion d'ontologie pour vérifier le contexte d'application d'un processus de fusion.

Deuxième partie

Travaux théoriques

Chapitre 5

Contrôle du système de fusion

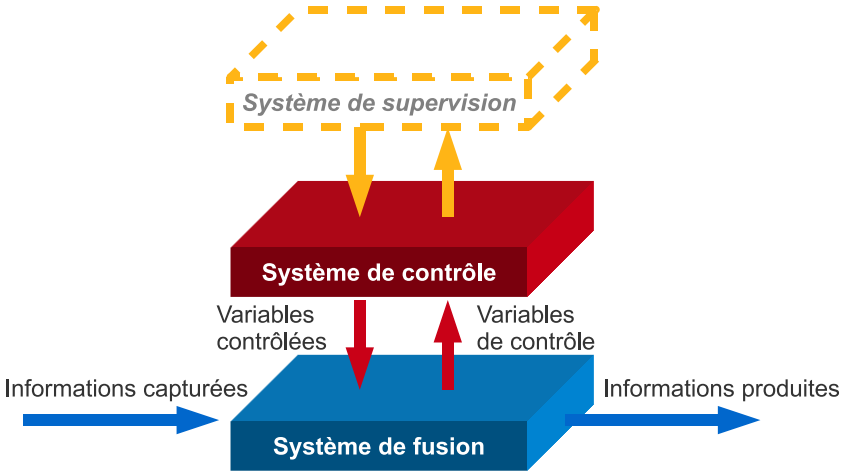


FIG. 5.1 – Architecture de contrôle à boucle fermée, appliquée à un système de fusion d'informations.

Pour débuter, rappelons dans un premier temps les quatre problématiques des SRFI que nous avons soulevées dans le Chapitre 2 : (1) la gestion de la volatilité des ressources, (2) l'adaptation de la puissance de calcul allouée au traitement, (3) le passage à l'échelle du système et (4) la réutilisation et la généricité du système. Par ailleurs, la synthèse que nous avons menée à la suite de l'état des travaux existants, révèle que ces problématiques sont actuellement partiellement résolues. Nous nous fixons donc pour objectifs d'apporter une solution à chacune d'elles au sein d'un même outil.

Alors que le Chapitre 2.2.2 présente les objectifs que nous nous sommes fixés, ce chapitre traite de l'organisation globale du système de contrôle que nous proposons. Celui-ci prend place à coté d'un second système, comme l'illustre la Figure 5.1, et se divise en cinq tâches de contrôle. La suite de ce chapitre est organisée selon trois paragraphes. Le Paragraphe 5.1 est dédié à la description fonctionnelle du système, complétée dans le Paragraphe 5.2 par une présentation du cycle de contrôle tel que nous le définissons. Le Paragraphe 5.3 définit les tâches de contrôle qui composent le système et que nous détaillons dans la suite du mémoire.

5.1 Système de contrôle

Le contrôle des systèmes est un domaine de recherche au même titre que la fusion d'informations. À la différence des modèles de fusion d'informations introduits dans le Chapitre 4, nous ne faisons pas état dans ce chapitre de tous les modèles de contrôle existants. Il nous est néanmoins indispensable de formuler le modèle de contrôle sur lequel nous nous basons [Bub05] :

- notre système, illustré par la Figure 5.1, est à boucle fermée ;
- plusieurs tâches de contrôle partiellement indépendantes, présentées dans le Paragraphe 5.3, sont appliquées au système de fusion ;
- et chaque nœud du système exécute une partie du système de contrôle et réagit en fonction de ses interactions avec les autres nœuds.

La dimension temporelle est également à préciser. En effet, notre système ne prétend pas effectuer de contrôle en temps réel et nous ne formulons pas de contraintes temporelles fortes sur le temps nécessaire à une prise de décision.

Buts du contrôle

Définition 5.1.1 (Contrôle d'un SFI)**.** *Notre étude des SFI a identifié deux axes de contrôle qui sont (1) le maintien ou l'amélioration des performances du service de fusion, et (2) la gestion de la qualité du résultat produit par le système.*

Le premier but du contrôle peut être atteint en modifiant par exemple le temps de réponse du système, sa disponibilité ou sa probabilité de tomber dans un état de panne. La qualité du résultat de fusion est au contraire liée à la précision du traitement et à l'incertitude ou la confiance du résultat.

Ces deux axes nécessitent un espace de mesure afin que le système de contrôle puisse prendre une décision et agir sur le SFI. Les mesures permettant la gestion des performances peuvent être effectuées directement sur le SFI. Les performances calculées à partir de ces mesures renseignent sur le comportement du système pour un processus de fusion donné.

Indépendamment de ce premier axe de contrôle, la qualité du résultat produit par le système nécessite une connaissance *a priori* du processus de fusion. En effet, la qualité des résultats (Paragraphe 6.3) est évaluée quelque soit la mise en œuvre du processus de fusion et nécessite donc l'expertise du concepteur du processus. Ainsi, la place du concepteur est primordiale dans cet axe du contrôle, tant lors de la description de l'évaluation de la qualité d'un résultat que lors du choix de l'action à appliquer pour l'améliorer.

5.2 Cycle de contrôle

5.2.1 Perception

Définition 5.2.1 (Perception du système de contrôle)**.** *Le système de contrôle perçoit la qualité du résultat d'un traitement grâce à l'évaluation d'un expert en fusion et perçoit également la performance du système de fusion grâce à des sondes installées dans celui-ci.*

Le premier indicateur, basé sur l'analyse d'un expert en fusion d'informations, peut être mesuré directement par l'expert grâce à des éléments de visualisation du résultat évalué. Cette mesure de la qualité d'un résultat peut être automatisée après que l'expert ait décrit un algorithme d'évaluation et l'ait implémenté dans le SFI.

Le second indicateur renseigne sur la performance du SFI. Celle-ci est automatiquement évaluée grâce à des mesures provenant de sondes internes au SFI. Ces sondes, que nous détaillons

dans le Paragraphe 6.7.2, renseignent par exemple le temps nécessaire à l'exécution d'une partie du processus ou à la quantité d'informations en attente dans le système.

5.2.2 Décision

Définition 5.2.2 (Décisions du système de contrôle)**.** *Les décisions sont prises suite à la détection d'une défaillance ou à la découverte d'une configuration plus performante et le système de contrôle évalue alors le bénéfice d'une reconfiguration.*

Dans le cadre de cette thèse, nous souhaitons automatiser les prises de décision du système de contrôle. Les données fournies en entrée du module de décision sont issues des surveillances conjointes de l'exécution du processus de fusion et des nœuds du système et les actions à appliquer sont envoyées à la partie du système en charge de l'implémentation du processus de fusion.

Les mécanismes de décision visant à améliorer la qualité des résultats du SFI sortent du cadre de l'étude de la thèse. De tels mécanismes sont néanmoins possibles et le concepteur du processus de fusion peut automatiser ce type d'amélioration par la description du processus de contrôle sous-jacent.

5.2.3 Action

Définition 5.2.3 (Actions du système de contrôle)**.** *Le système de contrôle permet trois actions sur le système de fusion : la modification de paramètres de fusion, la modification de la répartition de la configuration, et la modification de l'élément qui applique le traitement.*

Les actions disponibles permettent au système de contrôle du SFI d'en modifier la configuration pour l'améliorer selon les deux axes définis précédemment.

Lors de la description du processus de fusion d'informations, le concepteur peut définir des paramètres de contrôle pour chaque élément du processus. La mise à jour de ces paramètres, pendant l'exécution du processus, autorise un expert à ajuster le traitement de l'information et de surcroît la qualité des résultats produits.

Les performances du service de fusion sont améliorées lors de la modification de la répartition définie par la configuration du SFI et la modification de l'élément actif utilisé pour mettre en œuvre une partie du processus de fusion d'informations. Ainsi, une partie du traitement peut être affectée à des ressources différentes selon la configuration du système. De plus, notre approche permet la définition de plusieurs implémentations pour chaque élément du processus de fusion : par exemple une bibliothèque système ou l'appel à un service Web pour calculer une intégrale, le Chapitre 6 fournit plus de détails.

5.3 Tâches de contrôle

À présent que le système de contrôle commence à se dessiner, détaillons dans ce paragraphe les éléments qui le composent et qui nous permettent d'atteindre les objectifs tels que nous les avons énoncés dans le Chapitre 2.2.2.

Le système de contrôle et le système d'exécution reposent sur cinq tâches illustrées sur la Figure 5.2. La nature de ces tâches nous permet d'atteindre deux des objectifs, formulés dans le Chapitre 2.2.2 : la gestion de la volatilité des ressources (Paragraphe 2.3) et l'adaptation de la puissance allouée (Paragraphe 2.4). La suite de ce chapitre introduit chacune d'elles et plus de détails seront apportés par la suite dans les Chapitres 6 à 8. La répartition des tâches de contrôle, qui répond à notre objectif de passage à l'échelle du système, est présentée dans le Chapitre 9.

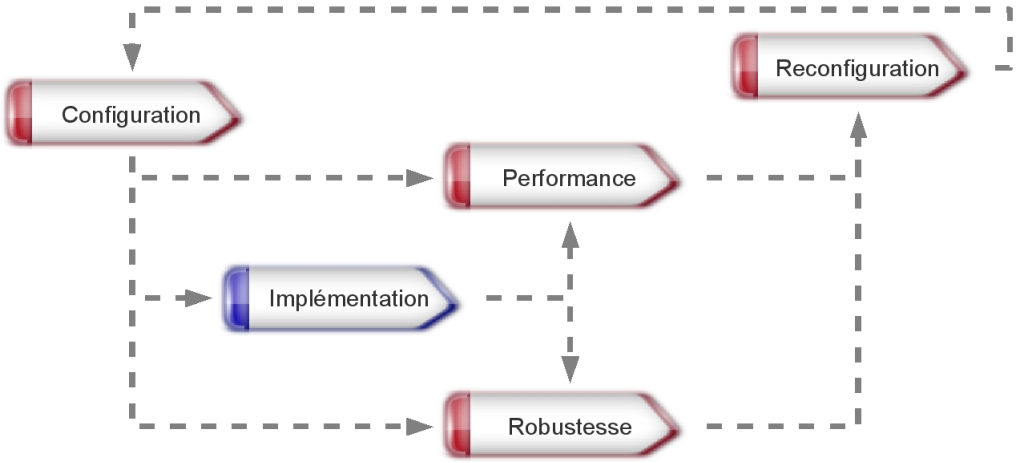


FIG. 5.2 – Détail de la boucle de contrôle : les flèches représentent la dépendance entre les tâches de contrôle.

5.3.1 Configuration

Définition 5.3.1 (Tâche *Configuration*). *La tâche de contrôle* Configuration *est chargée de l'acquisition de la description d'un nouveau processus de fusion à déployer et de la recherche d'une configuration compatible avec les ressources disponibles.*

Définition 5.3.2 (Configuration du SFI). *Une configuration du système définit l'affectation des éléments du processus de fusion sur les ressources du système et le choix du code exécutable qui applique le traitement décrit pour chaque élément du processus.*

La tâche de contrôle *Configuration* intervient dans la réalisation de la gestion de la volatilité des ressources et dans l'adaptation de la puissance de calcul allouée au traitement.

Après l'acquisition de la description d'un processus de fusion fournie par l'expert en fusion (le Chapitre 6 présente notre modèle de description) cette tâche de contrôle recherche une configuration possible. L'espace de recherche est construit d'une part grâce à des spécifications techniques fournies par l'administrateur du système, et d'autre part sur une vue des ressources automatiquement construite par le système. Cette vue est mise à jour par la tâche *Reconfiguration* qui statue sur la disponibilité d'une ressource. Cette dernière tâche de contrôle peut également requérir une nouvelle initialisation de la recherche à partir d'une configuration donnée lorsque celle-ci est évaluée plus performante.

Dès lors que le mécanisme de recherche est initialisé, une configuration est déployée et la tâche de contrôle *Configuration* la transmet à trois autres tâches de contrôle, comme l'illustre la Figure 5.2. La tâche de contrôle *Implémentation* déploie la configuration courante, alors que les tâches *Performance* et *Robustesse* sont chargées respectivement de l'analyse des performances du système et de la surveillance de l'exécution. Par ailleurs, les performances d'une configuration candidate peuvent également être analysées, et celle-ci peut alors être déployée.

5.3.2 Implémentation

Définition 5.3.3 (Tâche *Implémentation*). *La tâche de contrôle* Implémentation *évalue l'écart défini entre la configuration actuelle du système et celle qui doit être déployée, puis sélectionne le mode de transition en fonction de cet écart. Cette tâche de contrôle exécute le processus de fusion dès la fin du déploiement.*

La gestion de l'implémentation est nécessaire à la gestion de la volatilité des ressources et à l'adaptation de la puissance de calcul allouée au traitement.

Comme l'illustre la Figure 5.2, la tâche *Implémentation* dépend de la tâche *Configuration* qui lui fournit la configuration du système à déployer. Dès qu'une reconfiguration est déclenchée, cette tâche de contrôle évalue l'écart de configuration et sélectionne le mode de transition. Ces deux points sont détaillés dans le Chapitre 6.

Après le déploiement effectif d'une configuration, le traitement est appliqué aux informations capturées par le système et l'exécution du processus de fusion est surveillée par les tâches de contrôle *Performance* et *Robustesse*, également représentées sur le Figure 5.2. La nature de cette surveillance est présentée dans les paragraphes suivants.

5.3.3 Performance

Définition 5.3.4 (Tâche *Performance*). *La tâche de contrôle* Performance *construit un modèle d'analyse d'une configuration du système. Ce modèle est paramétré par des mesures issues du système et les indices de performance calculés sont transmis à une autre tâche de contrôle.*

La gestion de la performance est un tâche de contrôle qui intervient dans l'adaptation de la puissance de calcul allouée au traitement sous deux aspects. En effet, une configuration issue de l'espace de recherche est dans un premier temps traduite dans un modèle d'analyse. Les attributs de ce modèle sont basés sur une approximation *a priori* ou sur une série de mesures effectuées sur le système.

Ainsi, comme l'illustre la Figure 5.2, la tâche de contrôle *Performance* dépend de la tâche *Configuration* dont elle reçoit une configuration à analyser, et de la tâche *Implémentation* qui produit les mesures surveillées dans le système. Le système de contrôle surveille par exemple le temps nécessaire à une partie du traitement ou la quantité d'information en attente de traitement. Nous détaillons toutes ces mesures dans le Paragraphe 6.7.2.

Les indices de performance calculés à partir du modèle d'une configuration sont ensuite transmis à la tâche *Reconfiguration* qui décide de modifier l'espace de recherche ou d'initialiser la recherche dans la tâche *Configuration*.

5.3.4 Robustesse

Définition 5.3.5 (Tâche *Robustesse*). *La tâche de contrôle* Robustesse *établit une vue des ressources présentes dans le système et évalue leur disponibilité.*

Cette tâche de contrôle intervient dans la gestion de la volatilité des ressources.

Comme l'illustre la Figure 5.2, la tâche *Robustesse* reçoit d'une part la description de la configuration courante, issue de la tâche *Configuration*, et d'autre part des mesures effectuées sur le système *via* la tâche *Implémentation*. Ces mesures permettent l'élaboration d'une vue des ressources utilisées et disponibles, notamment les ressources ajoutées après la sélection de la configuration courante. Au contraire, si une ressource disparaît pendant l'exécution, des mécanismes de détection, présentés dans le Chapitre 8, notifient la tâche *Reconfiguration* afin que celle-ci déploie une nouvelle configuration compatible.

5.3.5 Reconfiguration

Définition 5.3.6 (Tâche *Reconfiguration*). *La tâche de contrôle* Reconfiguration *est l'élément de prise de décision du système. Elle gère l'évolution des ressources présentes dans le système et influe sur la recherche de configuration. Les reconfigurations du système sont déclenchées par cette tâche de contrôle.*

La gestion de la reconfiguration est nécessaire à la gestion de la volatilité des ressources et à l'adaptation de la puissance de calcul allouée au traitement.

Cette tâche de contrôle, notée *Reconfiguration* sur la Figure 5.2, reçoit des données agrégées basées sur les mesures effectuées sur le système. La prise de décision, introduite dans le Chapitre 8, intervient alors dans trois cas : (1) lorsque les performances d'une configuration ne sont pas celles prédites par le modèle d'analyse de la tâche *Performance*, (2) une autre configuration que celle courante a été évaluée plus performante par la tâche *Performance* ou (3) des ressources utilisées par la configuration courante sont notifiées indisponibles par la tâche *Robustesse*. Dans le dernier cas, la prise de décision est réduite à la modification de l'espace de recherche dans la tâche *Configuration* et au déploiement de la première configuration compatible. Dans les deux premiers cas, le déclenchement d'une reconfiguration est basé sur l'estimation du gain de performance entre la configuration courante et celle candidate.

5.4 Synthèse

Le contenu de ce chapitre représente notre premier apport. En effet, alors que nous avons identifié l'amalgame récurrent entre le contrôle d'un SRFI et la mise en œuvre d'un processus de fusion, nous proposons que ce contrôle soit explicité. Nous définissons ainsi deux sous-systèmes au sein d'un SRFI, respectivement pour le traitement effectif des informations et pour le contrôle du SRFI. Ce contrôle est par ailleurs réalisé selon deux axes que nous avons également identifiés. Le premier concerne le contrôle du processus de fusion. L'expert en fusion d'informations joue alors un rôle central dans l'évaluation des résultats intermédiaires ou finaux. De cette évaluation découle alors l'adaptation du processus de fusion. Le contrôle du système réparti mettant en œuvre le processus décrit alors le second axe de contrôle que nous avons identifié. Cet axe reprend des problèmes posés dans la communauté des systèmes répartis, et nous avons notamment sélectionné les problématiques de performances du système, ainsi que de robustesse. Les performances du SRFI sont dans notre contexte guidées par le temps de traitement d'une information depuis son entrée dans le système jusqu'à la production d'un résultat final. La gestion des défaillances des éléments du système, avec le maintien de l'exécution du processus et l'adaptation de la configuration du système concourent à l'amélioration de la robustesse du SRFI.

Chapitre 6

De la description à l'implémentation

Notre système de contrôle gère, entre autres, le déploiement et l'exécution d'un processus de fusion donné. Ainsi, parmi les tâches de contrôle que nous présentons dans le Chapitre 5, les tâches *Configuration* et *Implémentation* reposent sur les modèles présentés dans la suite de ce chapitre. Dans un premier temps nous délimitons dans le Paragraphe 6.1 le domaine des modèles de fusion d'informations que nous souhaitons gérer. Puis, dans le Paragraphe 6.2, nous présentons notre modèle de description d'un processus de fusion. Le modèle de déploiement d'un processus, avec les définitions d'une configuration du système et d'une mise en œuvre, exploite cette description d'un processus. Les Paragraphes 6.5 et 6.7 détaillent le fonctionnement des tâches *Configuration* et *Implémentation* au sein du système de contrôle.

6.1 Modèles de fusion visés

Le Paragraphe 4.2 a introduit les modèles de fusion d'informations dont le concepteur du SFI dispose pour concevoir les opérations appliquées aux informations. Tous ces modèles décrivent le traitement comme la composition d'opérations élémentaires où la nature de chaque opération découle du modèle de fusion. Notre approche propose la description de cette composition sous la forme d'un processus de fusion défini comme suit.

Processus de fusion

Définition 6.1.1 (Processus de fusion)**.** *Un processus de fusion est décrit par un graphe de flots de données où les nœuds représentent des éléments de traitement de l'information et où les arcs représentent les échanges de données.*

Notre approche restreint la description d'un processus de fusion d'informations à un graphe comme l'illustre la Figure 6.1. Ce graphe, acyclique, décrit le traitement des informations depuis

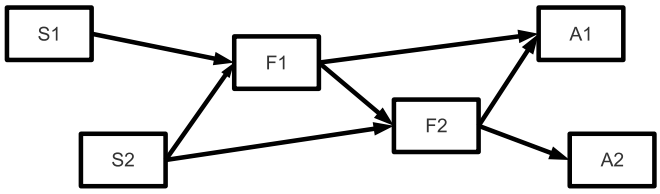


FIG. 6.1 – Le processus de fusion est décrit par un graphe de flots de données.

leur production par des sources, jusqu'à leur sortie du système au travers d'actionneurs (plus généralement des périphériques de sortie).

De part sa nature, un processus de fusion ne peut donc pas faire intervenir de bouclage comme ceux définis dans des processus de révision [BDPP06]. De même, les traitements d'informations qui proposent le raffinement de l'état de l'objet observé, comme par exemple lors de l'utilisation de filtres de Kalman [Sun06], sortent du cadre de notre approche. De tels traitements, aussi identifiés comme des processus de fusion d'analyse [Wal02], sont néanmoins autorisés à l'intérieur d'une opération élémentaire du traitement. Le concepteur doit par ailleurs veiller à la persistance de l'état interne au nœud de fusion car ce point reste indépendant de notre modèle.

Outre l'acyclicité du processus de fusion, la granularité de celui-ci impacte directement les choix des configurations. En effet, notre approche vise des domaines d'applications où les sources sont intrinsèquement distinctes des périphériques de sorties. Ainsi, plus la description d'un processus de fusion fera intervenir d'opérations élémentaires, plus les choix d'affectation des sous-parties du processus seront nombreux.

6.1.1 Contrôle du processus

Un des objectifs de cette thèse réside dans le contrôle du processus de fusion. Ce contrôle a lieu pendant le traitement et a pour but l'amélioration, ou le maintien, de la qualité du résultat issu du processus de fusion.

Comme nous l'avons entrevu précédemment, un mécanisme générique et autonome de contrôle du processus, si tant est qu'il soit possible, est hors de portée de nos travaux. Notre système doit néanmoins permettre la mesure de la qualité d'un résultat (intermédiaire ou final). Dès lors que la valeur de cet indicateur n'est pas satisfaisante, un mécanisme doit permettre la modification d'une partie du système afin de rectifier le traitement appliqué.

6.1.2 Qualité de la fusion

La fusion en théorie des possibilités et par des fonctions de croyances permet l'évaluation de la qualité du résultat du processus de fusion. Ainsi, la longueur de l'intervalle de confiance $[Bel(A), Pls(A)]$ est une mesure de l'ignorance que l'on a sur un événement A et son complémentaire. La qualité d'une décision issue d'un processus de fusion possibiliste est mesurée selon sa spécificité, c'est à dire selon un indice de précision de l'ensemble flou qui la représente.

Notre approche ne traite pas de l'amélioration autonome de cette qualité. Néanmoins, de par sa définition formelle, le concepteur du processus de fusion est en mesure d'automatiser cet axe du contrôle, grâce notamment à des boucles de contrôle agissant sur des paramètres du processus de fusion.

6.2 Description du processus de fusion

6.2.1 Élément de fusion

Définition 6.2.1 (Élément de fusion)**.** *Un élément de fusion est la vue algébrique ou algorithmique d'une opération élémentaire définie dans un des modèles présentés dans le Paragraphe 4.2.*

Le terme « fusion » désigne aussi bien l'amélioration d'une information que la composition de plusieurs informations. La production et la consommation sont deux cas spéciaux d'opérations élémentaires. Ainsi, un élément de fusion est la généralisation d'une fonction de fusion, d'une source d'informations et d'un actionneur. Ces deux derniers éléments étant des restrictions d'une fonction de fusion comme nous le présentons ci-dessous.

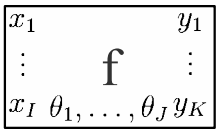


FIG. 6.2 – Représentation d'une fonction de fusion.

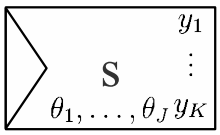


FIG. 6.3 – Représentation d'une source d'informations.

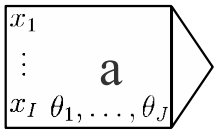


FIG. 6.4 – Représentation d'un actionneur.

Définition 6.2.2 (Fonction de fusion)**.** *Une fonction de fusion f est un ensemble de ports de communication. Cet ensemble forme trois ensembles disjoints : les ports entrées $\{x_1, \ldots, x_I\}$, les ports paramètres $\{\theta_1, \ldots, \theta_J\}$ et les ports résultats $\{y_1, \ldots, y_K\}$.*

Deux notations sont acceptées pour décrire une fonction de fusion :

$$f = \{x_1, \ldots, x_I, \theta_1, \ldots, \theta_J, y_1, \ldots, y_K\} \text{ ou } f_{(\theta_1,\ldots,\theta_J)}(x_1, \ldots, x_I) = (y_1, \ldots, y_K)$$

La notation condensée $f_\Theta(X) = Y$ suppose la définition des trois ensembles $\Theta = \{\theta_1, \ldots, \theta_J\}$, $X = \{x_1, \ldots, x_I\}$ et $Y = \{y_1, \ldots, y_K\}$. Une fonction de fusion est représentée par un rectangle dont trois des côtés décrivent les ports, comme l'illustre la Figure 6.2.

Définition 6.2.3 (Source d'information)**.** *Une source d'informations s est la restriction d'une fonction de fusion qui ne possède pas de ports d'entrée.*

Tout comme une fonction de fusion, une source d'informations accepte deux notations :

$$s\{\theta_1, \ldots, \theta_J, y_1, \ldots, y_K\} \text{ ou } s_{(\theta_1,\ldots,\theta_J)} = (y_1, \ldots, y_K)$$

La notation condensée $s_\Theta = Y$ suppose la définition des deux ensembles $\Theta = \{\theta_1, \ldots, \theta_J\}$ et $Y = \{y_1, \ldots, y_K\}$. Une source est représentée par un rectangle dont les ports d'entrées (côté gauche) sont masqués par un triangle, comme l'illustre la Figure 6.3.

Définition 6.2.4 (Actionneur)**.** *Un actionneur a est la restriction d'une fonction de fusion qui ne possède pas de ports de résultat.*

Sur le même modèle qu'une fonction de fusion ou une source d'information, un actionneur accepte deux notations :

$$a = \{x_1, \ldots, x_I, \theta_1, \ldots, \theta_J\} \text{ ou } a_{(\theta_1,\ldots,\theta_J)}(x_1, \ldots, x_I)$$

La notation condensée $a_\Theta(X)$ suppose la définition des deux ensembles $\Theta = \{\theta_1, \ldots, \theta_J\}$ et $X = \{x_1, \ldots, x_I\}$. Un actionneur, illustré par la Figure 6.4, est représenté par un rectangle dont les ports de résultats (côté droit) sont masqués par un triangle.

Propriété Un élément de fusion $e_\Theta(X) = Y$ vérifie les points suivants :
- $X \cap Y = \emptyset$;
- $X \cup Y \neq \emptyset$;
- Θ peut être vide.

Notation Pour faciliter l'identification d'un port de communication ou d'un vecteur de ports d'une fonction de fusion f, nous adoptons les notations suivantes :
- $f.x_1$ est un des ports d'entrées de la fonction f ;
- $f.y_1$ est un des ports de résultats de la fonction f ;
- $f.X$ est l'ensemble des ports d'entrées de la fonction f ;
- $f.Y$ est l'ensemble des ports de résultats de la fonction f.

$$\begin{gathered}\text{soit } f = \{x_1, \ldots, x_I, \theta_1, \ldots, \theta_J, y_1, \ldots, y_K\} \\ \text{alors } f.x_1 \Leftrightarrow x_1 \in \{x_1, \ldots, x_I, \theta_1, \ldots, \theta_J, y_1, \ldots, y_K\} \\ \text{et } f.X \Leftrightarrow \{x_1, \ldots, x_I\} \subset \{x_1, \ldots, x_I, \theta_1, \ldots, \theta_J, y_1, \ldots, y_K\}\end{gathered}$$

6.2.2 Processus

À présent que l'ensemble des éléments de fusion peuvent être décrits, attardons-nous à la composition de ces traitements élémentaires dans un processus de fusion.

Définition 6.2.5 (Lien). *Un lien $l \in L$ entre deux éléments de fusion e et e', issus de l'ensemble des éléments de fusion $\mathcal{F}$, est défini par le couple de ports qu'il connecte $l = (y, x')$ tel que $\exists (e, e') \in \mathcal{F}^2, y \in e.Y, x' \in e'.X$.*

Définition 6.2.6 (Processus de fusion). *Un processus de fusion d'informations $\mathcal{P} = (\mathcal{F}, L)$ décrit un graphe de flots de données discrétisées composé des éléments de fusion connectés par des liens.*

L'exemple illustré par la Figure 6.5 décrit un processus de fusion composé de quatre sources d'informations s_1, s_2, s_3, s_4, quatre fonctions de fusion f_1, f_2, f_3, f_4 et trois actionneurs a_1, a_2, a_3.

Validité du processus Un processus $\mathcal{P}$ vérifie les propriétés suivantes :

1. $\mathcal{P}$ est connexe ;
2. tout port d'entrées est connecté à un et un seul port de résultats ;
3. $\mathcal{P}$ est acyclique sur les ports d'entrées et de résultats.

La connexité du processus provient de la formulation d'une de nos hypothèses. En effet, notre approche traite de l'exécution d'un processus de fusion donné et la concurrence ainsi que les interactions de plusieurs processus font partie des ouvertures de nos travaux. Certains des indicateurs que nous définissons pour l'analyse des performances du système, comme la fréquence de production d'un résultat final, sont liés au processus de fusion déployé et il serait nécessaire de définir un opérateur d'agrégation pour plusieurs résultats finaux si cette hypothèse était relâchée. D'autres indicateurs, comme la charge CPU utilisée par le système, restent au contraire pertinents sous le relâchement de cette hypothèse.

La seconde hypothèse se traduit par l'explicitation du mécanisme de sélection lors de la conception du processus de fusion. En effet, pour un port d'entrée fixé, la politique de sélection des informations produites par deux sources connectées au même port sort de notre étude. Il est néanmoins possible d'en définir une, en l'explicitant par la conception d'un élément de fusion de sélection.

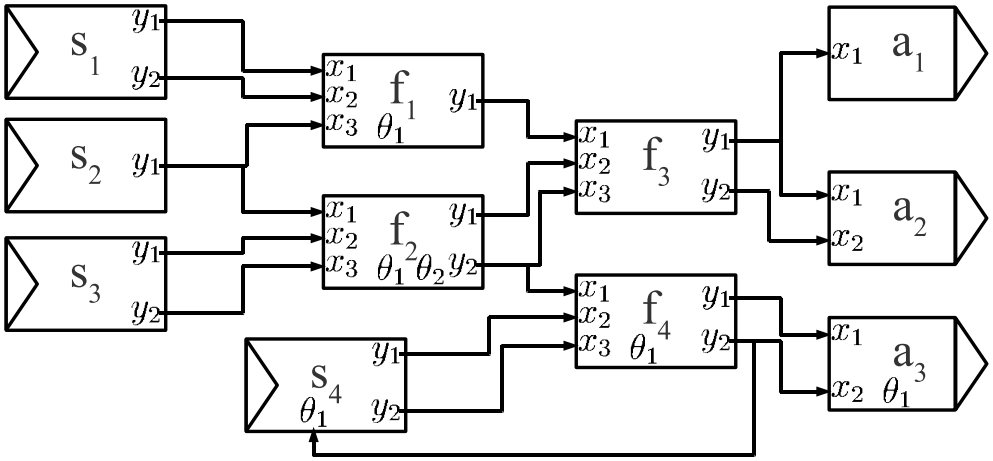


FIG. 6.5 – Exemple de description de processus de fusion d'informations.

L'acyclicité du processus de fusion est due à notre sémantique d'exécution (Paragraphe 6.6), car le lancement d'une exécution est bloquée tant qu'une partie du tuple d'entrées est non renseignée. Sans cette hypothèse des phénomènes d'inter-blocages seraient introduits lors de l'implémentation puis de l'exécution du processus.

Définition 6.2.7 (Ordre d'exécution). *La description du processus de fusion définit un ordre partiel d'exécution $O = (\mathcal{P}, <)$ sur l'ensemble des éléments de fusion.*

Cet ordre partiel est construit comme suit :

$$\forall l = (y, x') \in L, \exists (e, e') \in \mathcal{F}^2, y \in e.Y, x' \in e'.X \Rightarrow e < e',$$
$$\forall (e, e', e'') \in \mathcal{F}^3, e < e' \text{ et } e' < e'' \Rightarrow e < e'',$$
$$\forall (e, e') \in \mathcal{F}^2, e < e' \text{ et } e' < e \Rightarrow e = e'$$

Deux éléments de fusion comparables $e < e'$ définissent la relation « e est exécuté avant e' ». L'ensemble des éléments maximaux de O est équivalent à l'ensemble des actionneurs du processus de fusion et l'ensemble des éléments minimaux de O est équivalent à l'ensemble des sources d'informations.

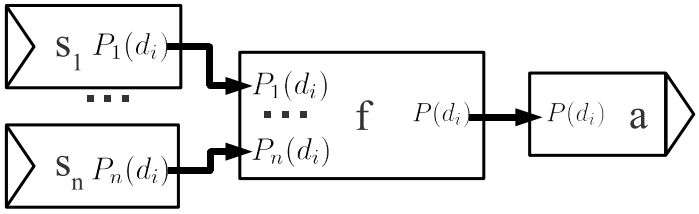
La dernière implication définie lors de la construction de l'ordre partiel d'exécution n'est pas utile dans notre approche. En effet, sous les hypothèses de validité du processus de fusion, nous restreignons la description du processus à des graphes acycliques. La condition $e < e'$ et $e' < e$ n'est donc jamais vérifiée.

Par ailleurs, certains processus de fusion définissent le cas particulier d'ordre total d'exécution lorsque la description du processus de fusion ne contient qu'une unique extension linéaire.

6.3 Compatibilité des modèles de fusion

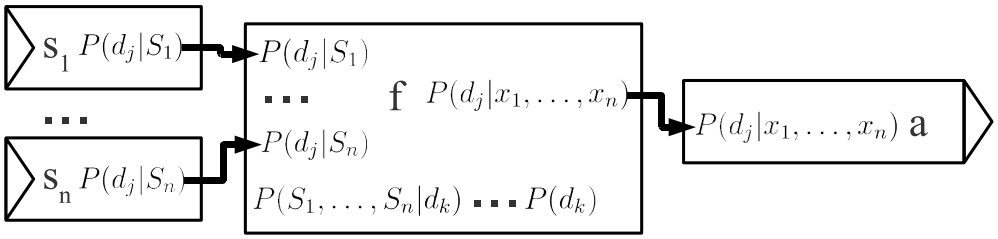
Dans ce paragraphe nous allons montrer comment décrire des fonctions de fusion basées sur les modèles du Chapitre 4. Nous discuterons de l'utilisation de réseaux de neurones à la fin de ce paragraphe.

Fusion probabiliste Ce premier exemple, illustrés par les Figures 6.6 et 6.7, présente la description des deux fonctions de fusion exprimées dans le modèle de fusion probabiliste. Les ports



$$P(d_i) = \sum_{j=0}^{n} \alpha_j \cdot P_j(d_i)$$

FIG. 6.6 – Fusion probabiliste non bayésienne.



$$P(d_j|x_1,\ldots,x_n) = \frac{P(S_1,\ldots,S_n|d_j)\cdot P(d_j)}{\sum_{k=1}^{n} P(S_1,\ldots,S_n|d_k)\cdot P(d_k)}$$

FIG. 6.7 – Fusion probabiliste bayésienne.

d'entrées représentent la même information dans les deux cas (la probabilité que l'objet observé appartienne à une classe donnée). La différence se situe au niveau des ports de paramètres. En effet, même s'il est possible de définir comme paramètre le poids d'une entrée dans la somme pondérée, la fusion probabiliste non bayésienne ne nécessite pas nécessairement ce type de ports. La fusion probabiliste bayésienne nécessite quant à elle l'estimation *a priori* des décisions, que le concepteur du processus doit fournir en paramètre de la fonction.

Le résultat issu de la fonction de fusion est dans les deux cas une probabilité. Une fois les données provenant des sources fusionnées, le processus de fusion peut décrire un niveau de décision probabiliste comme l'illustre la Figure 6.8.

Fusion en théorie des possibilités La description d'une fonction de fusion basée sur la théorie des possibilités s'effectue sur le même principe que dans le cas probabiliste non bayésien. La différence se fera lors de l'implémentation de la fonction de fusion et plus précisément au niveau des données manipulées. En effet, là où le modèle probabiliste définit une probabilité d'appartenance à une classe, la théorie des possibilités définit une distribution de probabilités sur l'ensemble des classes possibles. Cette distribution peut être discrétisée ce qui peut se traduire par une division des ports d'entrées, mais ce niveau de détails n'est requis que lors de l'implémentation.

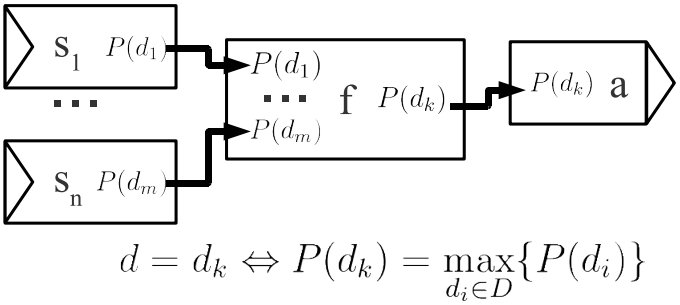


FIG. 6.8 – Décision probabiliste.

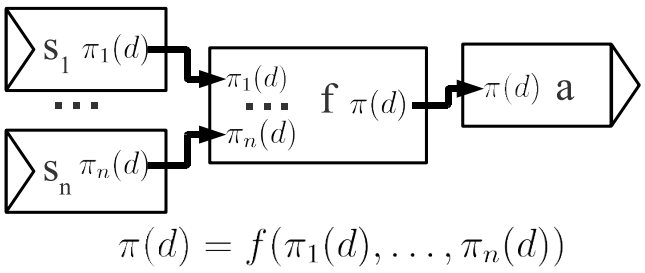


FIG. 6.9 – Fusion en théorie des possibilités : f est l'opérateur d'agrégation et d est un des éléments de l'ensemble de discernement.

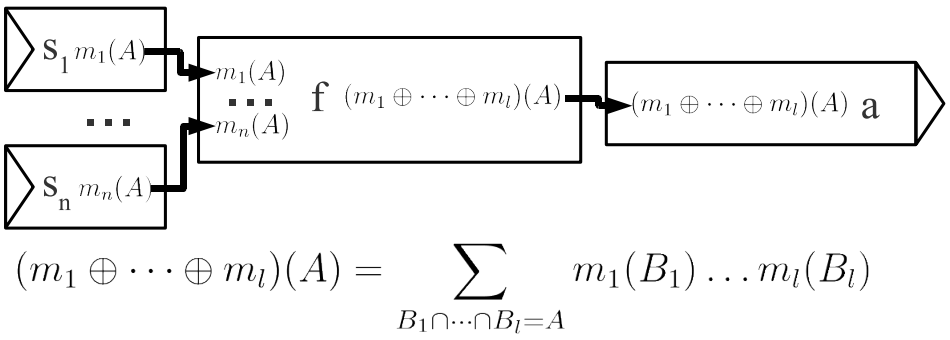


FIG. 6.10 – Fusion en théorie des fonctions de croyances : m_j la fonction de masse de la source s_j, la fusion [Sme90] s'effectue pour tout les sous-ensembles $A \subset D$.

Utilisation d'un réseau de neurones Même si notre approche n'est pas destinée au déploiement d'un réseau neuronal, il est néanmoins possible d'en définir un. Le premier point auquel le concepteur doit porter son attention concerne l'éventuelle boucle d'apprentissage définie dans le réseau. La traduction d'une telle boucle sera réalisée par des connexions sur les ports de paramètres des éléments de fusion.

Le choix de l'échelle de traduction est également important. En effet, tout un réseau peut être traduit par un élément de fusion ce qui implique le développement des connexions entre neurones au sein d'une implémentation (Paragraphe 6.6) : le processus de fusion décrit alors une composition de réseaux de neurones. Au contraire, un élément de fusion peut être le réceptacle d'un unique neurone et la description du processus de fusion définit alors les connexions entre les neurones. L'évaluation de la performance de ces choix de traduction sort de notre étude.

6.4 Modèle de déploiement du processus de fusion

Ce paragraphe présente les éléments nécessaires à la définition d'une configuration du SFI, illustrés par la Figure 6.11. Une configuration définit l'affectation de chaque élément de fusion sur un élément physique du système. Afin d'orienter la recherche d'une configuration, notre approche propose des attributs calculables sur les structures théoriques sous-jacentes.

6.4.1 Éléments d'une configuration

Définition 6.4.1 (Entité d'exécution ou implémentation)**.** *Une entité d'exécution définit le code exécutable, i.e. l'implémentation, d'un élément de fusion. Ce code exécutable est écrit par le développeur du processus en accord avec les indications fournies par le concepteur du processus de fusion.*

Plusieurs implémentations d'une même entité d'exécution peuvent être définies et celles-ci peuvent être répliquées dans le système. Cet élément de la configuration est le seul à dépendre du processus de fusion déployé.

Définition 6.4.2 (Cadre d'exécution)**.** *Un cadre d'exécution est un ensemble de ressources qui met en œuvre les mécanismes de contrôle du SFI. Il déploie les entités d'exécution qui lui sont affectées et propose les mécanismes de communication des informations partiellement traitées.*

En parallèle du déploiement d'une partie d'une configuration, un cadre d'exécution active les tâches de contrôle.

Définition 6.4.3 (Canal de communication et chemin)**.** *Un canal de communication est la connexion physique entre deux cadres d'exécution. Il est défini par l'adresse des cadres à chaque extrémité et est à double sens avec des propriétés symétriques. Une succession de canaux de communication définit alors un chemin dans le réseau.*

Définition 6.4.4 (Configuration du SFI)**.** *Une configuration $\mathcal{C} = (C_F, C_L, C_I)$ est définie par :*
- *C_F : l'affectation des éléments de fusion sur les cadres d'exécution,*
- *C_L : l'affectation des liens sur des chemins entre cadres d'exécution,*
- *C_I : le choix d'une implémentation sur les cadres d'exécution utilisés.*

6.4.2 Attributs d'une configuration

Configuration au niveau d'une entité d'exécution

Une configuration définit une implémentation à utiliser lors du déploiement. Si un élément de fusion est utilisé à plusieurs étapes du processus de fusion, autant d'implémentations différentes

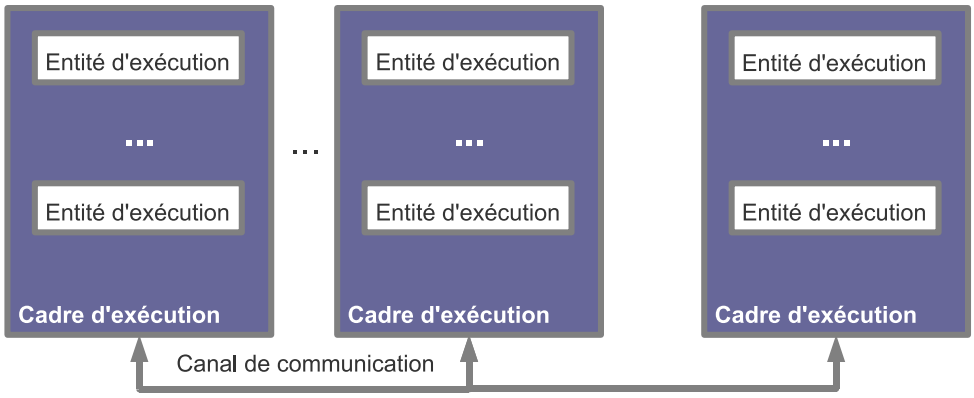


FIG. 6.11 – Organisation des éléments d'une configuration du système de fusion d'informations.

peuvent être déployées. Le choix d'une implémentation est néanmoins susceptible d'être changé, notamment si l'une d'elles est jugée défaillante.

La valeur des paramètres définis pour un élément de fusion fait également partie des attributs de la configuration du SFI. En effet, selon qu'un paramètre concerne le traitement ou l'implémentation, sa valeur est liée à l'élément de fusion ou à l'entité d'exécution. Alors que dans le premier cas la valeur du paramètre est définie par le concepteur du processus de fusion, le développeur de l'implémentation en est responsable dans le second cas. Ainsi, plusieurs valeurs du même paramètre peuvent être définies dans la même configuration, comme par exemple lors du choix d'une connexion pour une source d'informations.

Configuration au niveau d'un cadre d'exécution

La restriction d'une configuration à un cadre d'exécution définit :

1. les éléments de fusion que le cadre doit déployer,
2. les connexions pour transmettre les résultats intermédiaires.

D'autre part, une configuration définit également le chemin qu'un résultat partiel doit suivre dans le réseau. À ce titre, un cadre d'exécution est une étape dans un chemin et le cadre déploie alors des éléments induits par l'affectation de deux éléments de fusion sur d'autres cadres (nous détaillons la définition d'une étape dans la suite de ce chapitre).

Mesures de la configuration

Dans le but de pouvoir comparer deux configurations, nous proposons des mesures définies sur leur structure. Ainsi, en nous basant sur ces mesures, nous écartons les configurations peu performantes pour n'analyser plus finement que celles potentiellement « meilleures ». Notre approche propose les mesures suivantes :

- le nombre de cadres d'exécution qui consomment des informations produites par un cadre donné,
- le nombre d'éléments de fusion et de chemins affectés à un cadre donné,
- la longueur d'un chemin, en nombre d'étapes empruntées, pour relier deux éléments de fusion,
- le nombre de chemins qui partagent le même canal de communication,
- le diamètre, en nombre de canaux de communication, de la configuration.

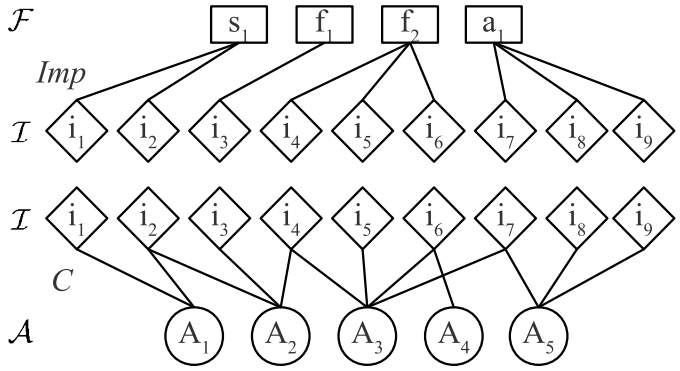


FIG. 6.12 – Ensemble des affectations possibles pour une configuration, vue étendue.

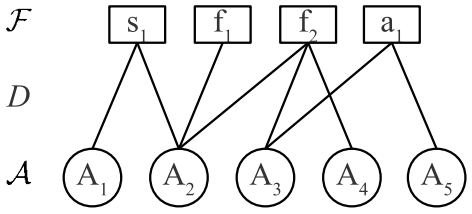


FIG. 6.13 – Ensemble des affectations possibles pour une configuration.

6.4.3 Modèle sous-jacent

Graphe bipartie

Nous avons choisi de baser notre algorithme de recherche de configurations sur deux graphes biparties B_C et B_{Imp}, regroupés sur la Figure 6.12, pour représenter la relation de compatibilité et celle d'implémentation.

Si l'on note $\mathcal{I}$ l'ensemble des implémentations et $\mathcal{A}$ l'ensemble des cadres d'exécution du système, alors $B_C = (\mathcal{I}, \mathcal{A}, C)$ définit le sous-ensemble des implémentations $I \subseteq \mathcal{I}$ compatibles avec un cadre d'exécution donné $A \in \mathcal{A}$. La relation $(i, A) \in C$ s'énonce alors « l'implémentation i peut être déployée sur le cadre d'exécution A ».

Une autre relation est également définie entre l'ensemble des éléments de fusion $\mathcal{F}$ et l'ensemble des implémentations $\mathcal{I}$. Ainsi, $B_{Imp} = (\mathcal{F}, \mathcal{I}, Imp)$ définit la relation d'implémentation entre les éléments de fusion et les implémentations possibles $(e, i) \in Imp$, laquelle s'énonce « l'élément de fusion e est implémenté par l'implémentation i ».

Le graphe bipartie B_D, illustré par la Figure 6.13, est une vue résumée des configurations possibles. Ainsi, $B_D = (\mathcal{F}, \mathcal{A}, D)$ représente les affectations possibles qui ne tiennent comptent que des cadres d'exécution disponibles. La relation $(e, A) \in D$ s'énonce « l'élément de fusion e peut être déployé sur le cadre d'exécution A ».

Pour une configuration $\mathcal{C} = (C_F, C_L, C_I)$ donnée, l'affectation des éléments de fusion C_F est modélisée par la couverture minimale et non redondante de $\mathcal{A}$ sur $\mathcal{F}$ dans B_D. C_I est alors le sous-ensemble de couples $\{(A, i)\} \subseteq \mathcal{A} \times \mathcal{I}$. Les Figures 6.14 et 6.15 illustrent ces deux affectations.

Matrice d'adjacence

L'ensemble des canaux de communication est représenté par une matrice d'adjacence N définie de $\mathcal{A} \times \mathcal{A}$ dans $\{0, 1\}$. Dans l'exemple de la Figure 6.16, les canaux de communication

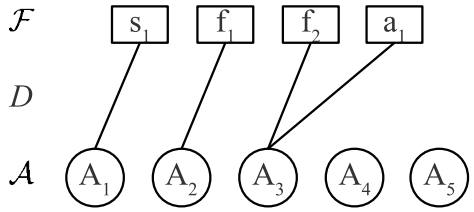


FIG. 6.14 – Exemple d'affectation d'éléments de fusion sur un ensemble des cadres d'exécution.

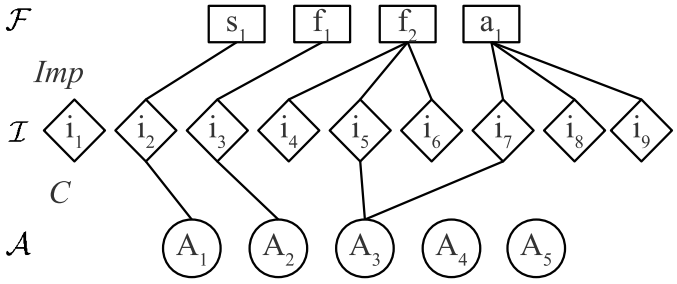


FIG. 6.15 – Exemple d'affectation d'éléments de fusion sur un ensemble des cadres d'exécution, vue étendue.

sont représentés par la matrice suivante :

$$N = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \end{pmatrix}$$ telle que $N_{ij} = 1 \Leftrightarrow$ les cadres a_i, a_j sont connectés.

Comme l'illustre la Figure 6.16, pour une configuration $\mathcal{C} = (C_F, C_L, C_I)$ donnée, C_L correspond à l'ensemble des chemins dans N entre tous les couples de nœuds du système qui exécutent deux fonctions successives dans l'ordre partiel d'exécution.

6.5 Gestion de la configuration

La tâche de contrôle *Configuration* définit trois types d'interactions illustrées sur la Figure 6.17 : (1) l'acquisition de la description d'un nouveau processus de fusion est réalisée avec le concepteur du processus de fusion, (2) la modification de l'espace de recherche de configuration est guidée par la tâche de contrôle responsable de la reconfiguration du système et (3) le déploiement d'une nouvelle configuration implique l'envoi d'un message de contrôle en direction des tâches *Implémentation*, *Performance* et *Robustesse*.

Nouveau processus

Lorsqu'un processus de fusion doit être déployé, le concepteur fournit sa description au système de contrôle. En amont du déploiement, il est nécessaire de définir une configuration du SFI qui implémente le processus souhaité. Ainsi, la tâche *Configuration* construit l'ensemble des configurations possibles en tenant compte de la disponibilité de chaque ressource et sélectionne la

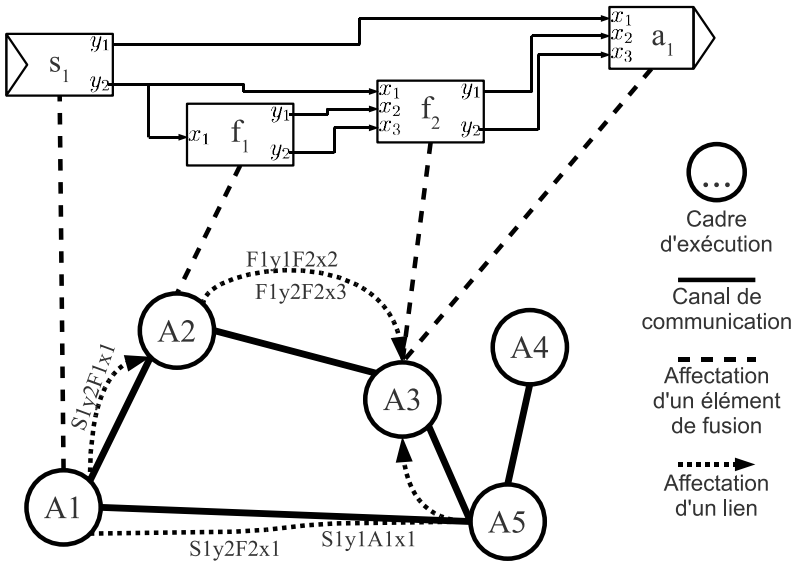


FIG. 6.16 – Exemple de configuration : affectation des éléments du processus de fusion aux ressources disponibles.

première compatible. En parallèle du déploiement de cette configuration, le parcours de l'ensemble des configurations possibles s'effectue en tâche de fond. Si cette recherche aboutit à une configuration candidate, celle-ci est transmise à la tâche *Performance* pour être préalablement évaluée.

Mise à jour des attributs du système

Cette mise à jour, décidée par la tâche *Reconfiguration*, modifie l'espace des configurations possibles, comme lors de l'ajout ou de la suppression d'un cadre d'exécution ou d'une relation de compatibilité. La mise à jour peut également avoir lieu sur la matrice d'adjacence du réseau.

Si les modifications apportées à l'espace de recherche invalident la configuration actuellement déployée, la recherche de configuration est réinitialisée à la première configuration compatible.

Par ailleurs, la tâche *Reconfiguration* peut également réinitialiser la recherche à partir d'une configuration évaluée plus performante que celle actuellement déployée.

Déploiement d'une nouvelle configuration

Le déploiement d'une configuration a lieu dès qu'une configuration est évaluée plus performante que la configuration courante. La description de cette configuration est alors transmise à trois autres tâches de contrôle (*Implémentation*, *Performance* et *Robustesse*).

La tâche *Implémentation* déploie effectivement la configuration sur le système d'exécution pour appliquer le traitement décrit dans le processus de fusion. Les deux autres tâches de contrôle qui reçoivent la description d'une configuration, c.f. la Figure 6.17, construisent des modèles d'analyse du système pour le surveiller. Ces modèles sont mis à jour comme nous l'expliquons dans les Chapitres 7 et 8.

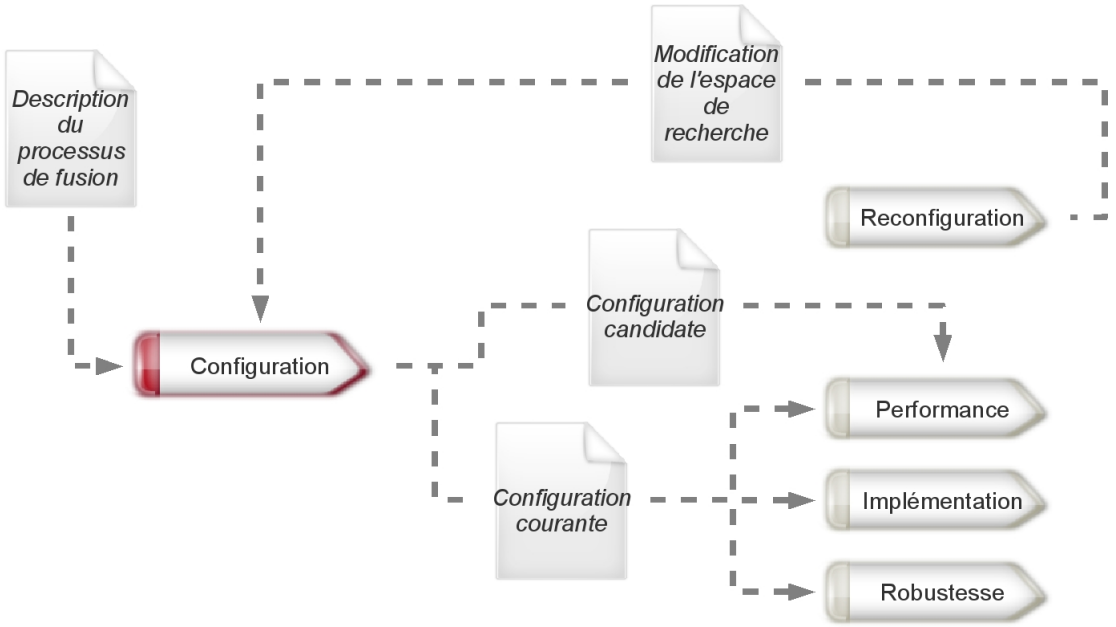


FIG. 6.17 – Tâche de contrôle liée à la configuration.

6.6 Implémentation du processus de fusion

Une implémentation du processus de fusion est définie comme le code exécutable qui permet la lecture des sources d'informations, puis transmet les résultats partiels entre les éventuels niveaux de traitement pour enfin produire un résultat final *via* des actionneurs.

Le processus de fusion ainsi implémenté est supposé constant pendant l'exécution. L'implémentation peut quant à elle évoluer, conformément aux adaptations de la configuration du système.

6.6.1 Implémentation d'une configuration

L'implémentation d'une configuration est la mise en œuvre effective du processus de fusion. À cette fin nous définissons la traduction de chaque élément d'une configuration par un ou plusieurs éléments exécutables.

Entité d'exécution

À ce niveau de description du système, nous devons étendre la définition d'une entité d'exécution pour tenir compte de la sémantique d'exécution que nous adoptons. En effet, l'exécution du code suit trois étapes :

1. la lecture d'un vecteur d'informations d'entrée ;
2. la lecture d'un vecteur de paramètres ;
3. l'exécution du code de l'implémentation.

Les vecteurs lus supposent qu'aucune de leurs composantes n'est vide. Si les informations d'entrées sont issues d'éléments de fusion, la valeur des paramètres est soit calculée par élément de fusion (grâce à une boucle de contrôle), soit fixée par un élément indépendant du système. De là, nous posons qu'un paramètre d'un élément de fusion s'accompagne toujours d'une valeur

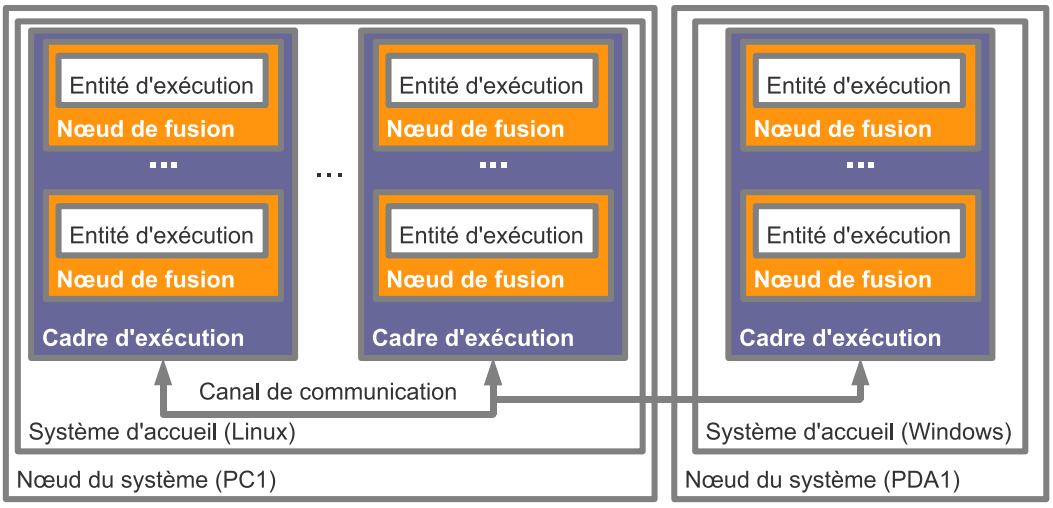


FIG. 6.18 – Organisation des éléments d'une implémentation de processus de fusion.

par défaut établie soit par le concepteur du processus de fusion, soit par le développeur de l'implémentation. Concernant le vecteur d'informations de données, le système est dans un état d'attente tant qu'une des composantes est vide.

Lors de l'exécution du code de l'implémentation, notre système capture les éventuelles erreurs non gérées par le développeur de l'implémentation et notifie le système de contrôle de la nécessité d'une reconfiguration. Si aucune erreur ne survient, l'exécution du code de l'implémentation se termine par la production d'un vecteur d'informations de résultats dont aucune composante ne doit être vide.

Nœud de fusion

Définition 6.6.1 (Nœud de fusion)**.** *Un nœud de fusion contrôle l'entité d'exécution. Cette enveloppe, est composée de l'entité d'exécution qu'elle contrôle, d'une file d'attente d'informations pour chaque port d'entrée, d'un vecteur de paramètres et de sondes logicielles.*

Un nœud de fusion, représentée sur la Figure 6.19, est un élément générique que nous introduisons dans le but de contrôler l'entité d'exécution. Il permet la gestion de l'exécution d'une entité d'exécution en utilisant le premier élément des files d'attente comme composante du vecteur de données défini précédemment.

Au-delà du stockage des informations, le nœud de fusion propose une partie du système de contrôle. Ainsi, les mesures issues des sondes internes au système (présentées dans le Paragraphe 6.7.2) sont agrégées dans les niveaux de contrôle supérieurs et plusieurs actions peuvent être appliquées sur le nœud.

Un nœud de fusion propose deux types d'actions de contrôle : l'échange d'implémentations et la mise à jour de paramètres. La dernière action intervient suite à l'évaluation de la qualité du résultat produit par l'entité d'exécution. La nouvelle valeur du paramètre, fournie par le concepteur du processus, est mise à jour puis utilisée lors de l'exécution du prochain vecteur d'informations de données. En effet, nous supposons que la valeur d'un paramètre reste constante pendant une exécution. La seconde action, l'échange d'implémentation, est nécessaire quand une autre implémentation de l'élément d'exécution est jugée plus performante. Dans ce cas, le traitement reste identique pour les deux implémentations et les données en attente peuvent être traitées.

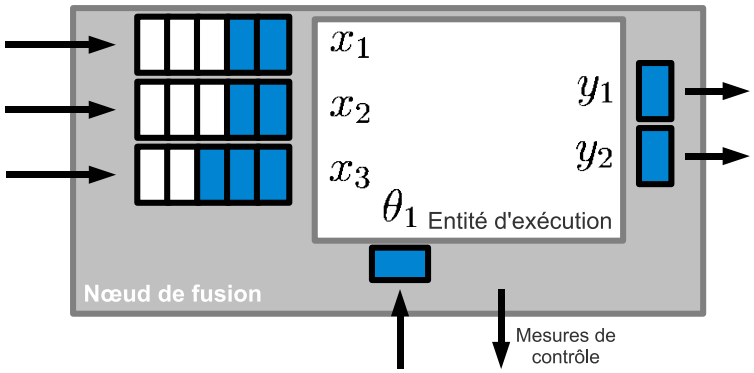


FIG. 6.19 – Composition d'un nœud de fusion.

Étape

Définition 6.6.2 (Etape). *Une étape est l'implémentation totale ou partielle d'un lien entre deux éléments de fusion.*

Ainsi, une configuration définit pour chaque lien un chemin qui empreinte les canaux de communication reliant les cadres d'exécution. Ce chemin, de longueur positive ou nulle, peut effectuer plusieurs *étapes* entre deux nœuds de fusion et nous posons qu'une étape locale implémente un chemin de longueur nulle. L'implémentation d'un chemin est au contraire une succession d'étapes distantes, affectées à autant de canaux de communication. Dans l'exemple illustré par la Figure 6.16, le lien entre la source d'information S_1 et la fonction de fusion F_2, respectivement affectées aux cadres A_1 et A_3, est implémenté par deux étapes de A_1 vers A_5 puis de A_5 vers A_3.

Cadre d'exécution

Un cadre d'exécution est composé de deux conteneurs aux rôles différents. Le premier conteneur correspond au système d'exécution et héberge les nœuds de fusion déployés sur le cadre alors que le second correspond au système de contrôle et sert au contrôle du premier. Ces deux sous-systèmes sont présentés dans le Chapitre 5 et nous détaillons ce choix technique dans la Partie « Mise en œuvre » du mémoire. Pour des raisons de clarté, nous avons simplifié la Figure 6.18 où les deux sous-systèmes ne sont pas représentés. En effet, ce chapitre se focalise sur la définition des éléments qui implémentent le processus de fusion et non l'ensemble du système de contrôle.

Système d'accueil et nœud du système

Définition 6.6.3 (Système d'accueil). *Un système d'accueil héberge au moins un cadre d'exécution.*

Nous proposons un système développé dans le cadre de plates-formes OSGi [OSG], car la plupart des systèmes d'exploitation proposés sur les systèmes que nous visons (c.f. Chapitre 2.2.2) sont susceptibles d'être utilisés. De plus, si la puissance disponible le permet, plusieurs cadres d'exécution peuvent être installés sur le même système d'accueil, comme l'illustre la Figure 6.18.

Définition 6.6.4 (Nœud du système). *Un nœud du système héberge au moins un système d'accueil et son architecture correspond à celles que nous avons présentées dans le Chapitre 2.2.2.*

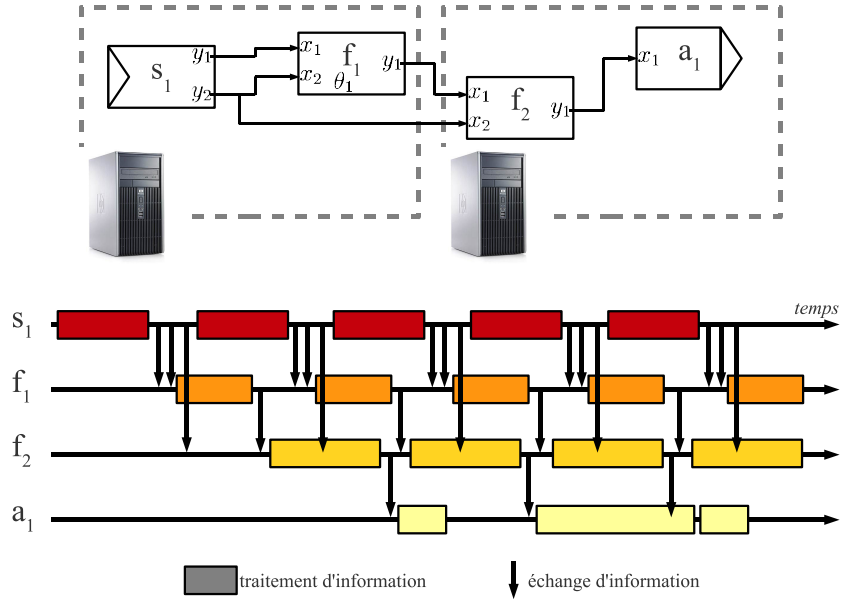


FIG. 6.20 – Exemple d'interactions de nœuds de fusion pendant une exécution.

6.6.2 Interactions des éléments

Ce paragraphe présente les éléments d'une implémentation dans un repère temporisé. Nous détaillons dans les deux sous-paragraphes suivants les interactions entre ces éléments, en nous basant sur les exemples d'exécution et de transition des Figures 6.20 et 6.21.

Pendant une exécution

Notre sémantique d'exécution définit trois verrous qui limitent le lancement du traitement d'un vecteur de données. Ainsi, le premier verrou concerne le vecteur d'informations à traiter. Ce verrou est libéré dès lors qu'un vecteur est prêt à être traité, i.e. que toutes ses composantes sont renseignées.

Le second verrou renseigne sur l'état des nœuds de fusion connectés aux ports de résultats. En effet, notre approche lance le traitement d'un vecteur de données uniquement si tous les nœuds consommateurs sont prêts à recevoir le résultat. Les résultats issus du traitement sont en outre transmis port par port, comme le suggère la Figure 6.20. L'envoi d'un couple d'informations entre deux nœuds de fusion nécessite donc deux transferts. De plus, le résultat issu d'un seul port nécessite également deux transferts d'information s'il est consommé par deux nœuds de fusion. Par ailleurs, l'arrivée d'informations sur les ports de données n'a pas d'influence sur l'exécution actuellement en cours, comme l'illustre la Figure 6.20.

Le dernier verrou activable représente l'état d'exécution du nœud de fusion. Ce verrou est le plus souvent ouvert et n'est fermé que dans le cas d'une reconfiguration du cadre d'exécution.

Pendant une transition

L'exemple de reconfiguration illustré par la Figure 6.21 propose le déplacement du nœud de fusion f_2 entre deux cadres d'exécution, ici installés sur deux nœuds du système pour des raisons de simplicité. Dans cet exemple, l'écart de configuration correspond à un déplacement et les nœuds de fusion connectés au nœud déplacé vont être affectés (le Paragraphe 6.7 définit les différents écarts de configuration possibles).

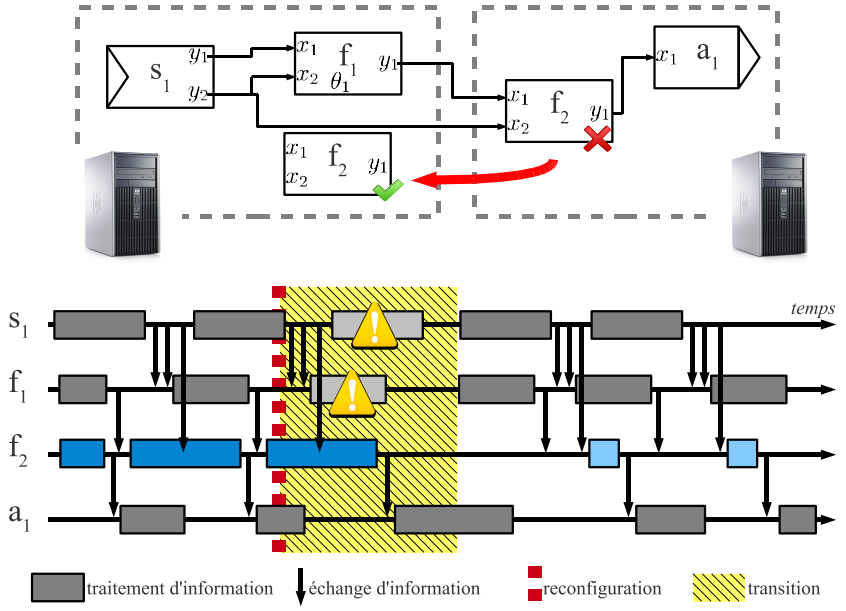


FIG. 6.21 – Exemple d'interactions de nœuds de fusion pendant une transition.

Une reconfiguration (ligne discontinue sur la Figure 6.21) peut par exemple être déclenchée suite à la découverte d'une configuration plus performante. À partir de cet instant, le déplacement du nœud de fusion passe par les étapes suivantes sur les cadres dont la configuration est modifiée :

- attendre la fin de l'exécution courante de f_2 et le transfert des résultats,
- verrouiller l'exécution des nœuds d'exécution qui produisent des informations pour f_2 : les traitements de s_1 et f_1 ne sont pas lancés pendant la transition, celui de a_1 n'est pas affecté,
- destruction du nœud de fusion déplacé ainsi que des étapes qui implémentent les liens connectés aux ports d'entrées et de résultats de f_2,
- création du nœud de fusion au nouvel emplacement et création des étapes qui implémentent les liens précédents,
- relâchement des verrous d'exécution de s_1 et f_1.

6.7 Gestion de l'implémentation du processus

La tâche de contrôle en charge de l'implémentation du processus de fusion s'articule autour de deux rôles. Le premier rôle attribué à cette tâche de contrôle porte sur l'interface indispensable au système de contrôle pour établir une vue du SFI. Des sondes installées dans l'implémentation du processus permettent la prise d'un ensemble de mesures que nous détaillons dans le Paragraphe 6.7.2.

L'autre rôle de la tâche *Implémentation* concerne la transition entre deux configurations pendant la reconfiguration de l'implémentation. Cette tâche de contrôle est répartie sur l'ensemble du SFI ce qui permet une gestion locale de cette transition dont la politique dépend de l'écart de configuration que nous définissons dans le Paragraphe 6.7.1.

6.7.1 Reconfiguration de l'implémentation

Écart de configuration

Étant données deux configurations $\mathcal{C}_1$ et $\mathcal{C}_2$ telles que définies dans le Paragraphe 6.4.3, l'écart de configuration, noté $\Delta(\mathcal{C}_1, \mathcal{C}_2) = (\delta_F, \delta_L, \delta_I)$, correspond aux modifications appliquées à la configuration $\mathcal{C}_1$ pour avoir $\mathcal{C}_2$. Les modifications possibles, non exclusives, sont :

- une mise à jour de l'affectation des éléments de fusion, notée δ_F,
- une mise à jour de l'affectation des liens, notée δ_L,
- une mise à jour du choix des implémentations utilisées, notée δ_I.

Par ailleurs, la mise à jour de la valeur d'un paramètre d'un élément de fusion ne constitue pas une modification de la configuration.

Calcul de δ_F Soit $Ci_F = \{(e, A)\} \subseteq D_i$ l'affectation des éléments de fusion dans chaque configuration avec D_i l'ensemble des affectations possibles dans $Bi_D = (\mathcal{F}_i, \mathcal{A}_i, D_i)$. Les éléments de fusion déplacés entre $\mathcal{C}_1$ et $\mathcal{C}_2$ sont donc :

$$\delta_F = (C1_F \cup C2_F) - (C1_F \cap C2_F)$$

L'ensemble des éléments de fusion déplacés entre deux configurations n'est exploité que pour un agent donné. En effet, le but est d'être en mesure de calculer quels sont les éléments ajoutés et supprimés sur un cadre d'exécution.

Pour un cadre d'exécution A donné, le calcul des nouveaux éléments de fusion affectés $A_F(A) \subseteq \mathcal{F}$ est :

$$A_F(A) = \{e \in \mathcal{F} | (e, A) \in \delta_F \cap C2_F\}$$

Respectivement les éléments de fusion supprimés $S_F(A) \subseteq \mathcal{F}$ sont :

$$S_F(A) = \{e \in \mathcal{F} | (e, A) \in \delta_F \cap C1_F\}$$

Calcul de δ_L Soit $Ci_L = \{(l, A \dots A_k \dots A') | l \in L, A_k \in \mathcal{A}\} \subseteq 2^N$ l'ensemble des chemins empruntés par la configuration $\mathcal{C}_i$ pour implémenter les liens $l \in L\}$. Les chemins modifiés entre $\mathcal{C}_1$ et $\mathcal{C}_2$ sont donc :

$$\delta_L = C1_L - (C1_L \cap C2_L)$$

De même que le calcul de δ_F n'est exploité que pour un cadre donné, l'ensemble des cadres d'exécution dont une étape a été modifiée n'est utile que pour un cadre donné. Soit A un cadre d'exécution, les ensembles des étapes supprimées $S_L(A)$ et ajoutées $A_L(A)$ sont déterminés par :

$$\begin{gathered}\text{soit } (A, A') \in \mathcal{A} \times \mathcal{A}, (A, A') \neq (A, \emptyset), \\ S_L(A) = \{(A, A') | \exists l \in L, (l, \dots AA' \dots) \in \delta_L \cap C1_L \vee (l, \dots A'A \dots) \in \delta_L \cap C1_L\}, \\ A_L(A) = \{(A, A') | \exists l \in L, (l, \dots AA' \dots) \in \delta_L \cap C2_L \vee (l, \dots A'A \dots) \in \delta_L \cap C2_L\}\end{gathered}$$

Calcul de δ_I Soit $Ci_I = \{(A, i)\} \subseteq \mathcal{A} \times \mathcal{I}$ l'ensemble des implémentations utilisées pour le déploiement de la configuration $\mathcal{C}_i$. Dans ce type de modification nous nous intéressons aux échanges d'implémentations effectués sur un cadre d'exécution donné A. L'ensemble des tels échanges est donc :

$$\begin{gathered}\text{posons } B_{imp} = (\mathcal{F}, \mathcal{I}, Imp), \\ \text{soient } (A, i, i') \in \mathcal{A} \times \mathcal{I} \times \mathcal{I}, (A, i) \in C1_I, (A, i') \in C2_I, \\ \delta_I = \{(A, i, i') | \exists e \in \mathcal{F}, (e, i) \in Imp, (e, i') \in Imp\}\end{gathered}$$

Pour un cadre d'exécution donné A, l'ensemble des implémentations à échanger, noté $Sw(A)$, est alors :

$$Sw(A) = \{(i, i') \in \mathcal{I}^2 | (A, i, i') \in \delta_I\}$$

Mécanismes de transition

Dès que l'écart de configuration est établi entre la configuration courante et celle à déployer, le système de contrôle sélectionne une des transitions suivantes :

- sw est un échange d'implémentation,
- st est une modification d'étape,
- as est une modification d'affectation.

Ces mécanismes de transition sont tous définis pour un cadre d'exécution donné. De plus, les effets de ces mécanismes sont dépendants et suivent la règle d'implication suivante :

- $as \Rightarrow st \wedge sw$

Échange d'implémentation Ce mécanisme de transition s'applique à un couple (i, i') tel que i et i' sont respectivement l'implémentation courante et la nouvelle implémentation du même élément de fusion.

Pour un écart de configuration $\Delta(\mathcal{C}_1, \mathcal{C}_2)$ donné, ce type de transition est appliqué sur le cadre d'exécution A pour tout couple $(i, i') \in Sw(A)$.

Ce mécanisme de transition ne modifie pas les informations partiellement traitées en attente dans le nœud de fusion, ni les étapes connectées. De plus, les nœuds de fusion qui produisent des informations à destination du nœud modifié peuvent effectuer leur traitement, i.e. le nœud modifié reste disponible. En revanche, le traitement est suspendu pendant la transition de ce nœud de fusion.

Modification d'étape Ce mécanisme de transition est déclenché suite à la modification de l'affectation d'un lien sur le réseau sans changement d'affectation des éléments de fusion ou suite au changement de l'affectation d'un élément de fusion. La modification d'étape est appliquée sur un cadre d'exécution donné A et s'applique sur un de ses canaux de communication.

Pour un écart de configuration $\Delta(\mathcal{C}_1, \mathcal{C}_2)$ donné, une étape sera créée pour tout couple $(A, A') \in A_L(A)$ et une étape sera supprimée pour tout couple $(A, A') \in S_L(A)$.

La suppression d'une étape implique l'arrêt du traitement du nœud de fusion connecté par un de ses ports de résultats. Dès que les modifications apportées à un chemin sont appliquées, le du nœud de fusion connecté par un de ses ports de résultats reprend son traitement.

Modification d'affectation Une modification d'affectation s'applique sur deux cadres d'exécution : celui dont l'affectation d'un nœuds de fusion est supprimée et celui où elle est ajoutée.

Pour un écart de configuration $\Delta(\mathcal{C}_1, \mathcal{C}_2)$ donné, un nœud de fusion sera déployé sur le cadre d'exécution A pour tout élément de fusion $e \in A_F(A)$ et l'implémentation sélectionnée est donnée dans la nouvelle configuration $\mathcal{C}_2$. Respectivement, un nœud de fusion sera supprimé pour tout élément de fusion $e \in S_F(A)$.

Une telle transition, illustrée par la Figure 6.21, a plusieurs effets. Ainsi, les étapes connectées au nœud de fusion modifié devront être ajoutées ou supprimées selon le cas - ceci implique l'arrêt du traitement pour les nœuds de fusion précédant le nœud reconfiguré. De plus, les informations partiellement traitées ne sont pas transférées pendant la transition car la sauvegarde de données redondantes et les mécanismes de restauration après panne sortent du cadre de nos travaux.

6.7.2 Mesures du système de fusion

En parallèle de l'application d'actions sur l'implémentation du processus de fusion, la tâche de contrôle *Implémentation* (Figure 6.22) permet l'élaboration d'un premier niveau de contrôle. En effet, des sondes installées dans le système de contrôle, illustrées par la Figure 6.23, fournissent

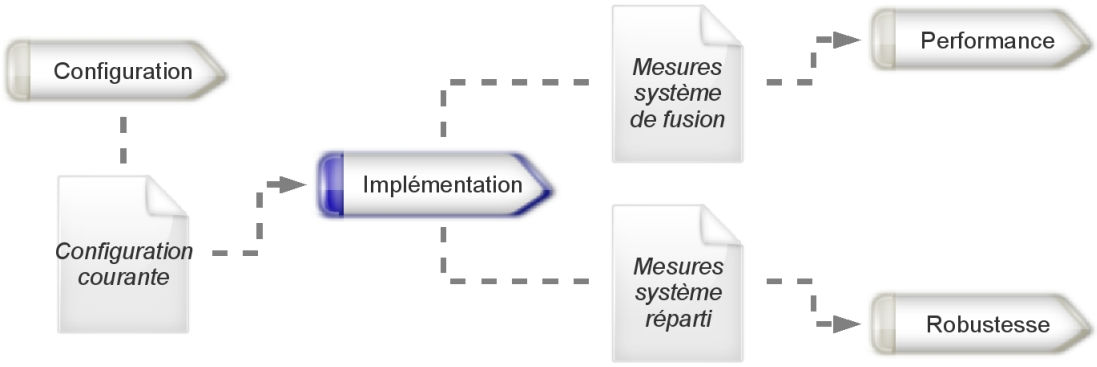


FIG. 6.22 – Tâche de contrôle liée à l'implémentation.

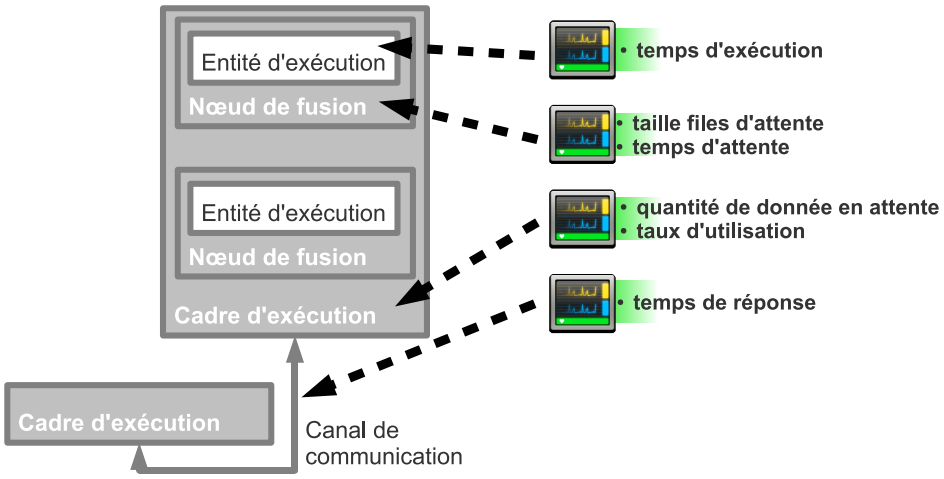


FIG. 6.23 – Variables de contrôle issues de l'implémentation du processus de fusion.

des mesures pour l'analyse des performances du SFI et pour la gestion de la dynamicité des ressources.

Entité d'exécution Toutes les mesures effectuées à ce niveau sont utilisées dans l'évaluation des performances d'une configuration du SFI. Nous avons identifié deux mesures possibles que nous exploitons en partie : le temps d'exécution et l'espace mémoire.

Le temps d'exécution d'une implémentation, représenté sur la Figure 6.24, mesure l'intervalle de temps entre l'instant où le traitement est lancé et l'instant où le résultat est disponible. Cette mesure est exploitée directement par la tâche de contrôle *Performance* lors de la mise à jour du temps de franchissement de l'élément du modèle qui correspond à l'implémentation. Par ailleurs, le temps d'exécution est également surveillé par la tâche *Robustesse* qui détecte un délai d'exécution anormal, au regard des exécutions précédentes. L'espace mémoire utilisé par l'entité d'exécution est une autre mesure possible que nous n'avons pas retenue. L'exploitation de cette mesure serait utile si des données devaient être stockées dans l'implémentation d'un élément de fusion. Or nous avons choisi de restreindre le rôle d'une entité d'exécution à l'application d'un traitement.

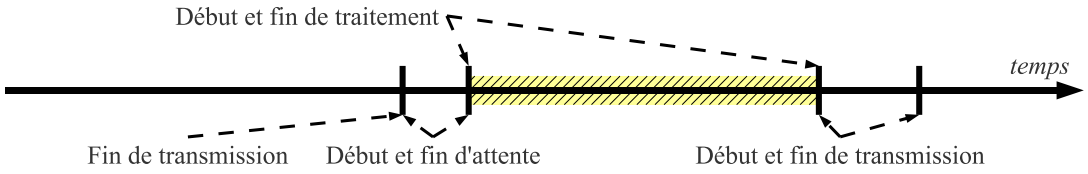


FIG. 6.24 – Temps d'exécution d'une entité d'exécution.

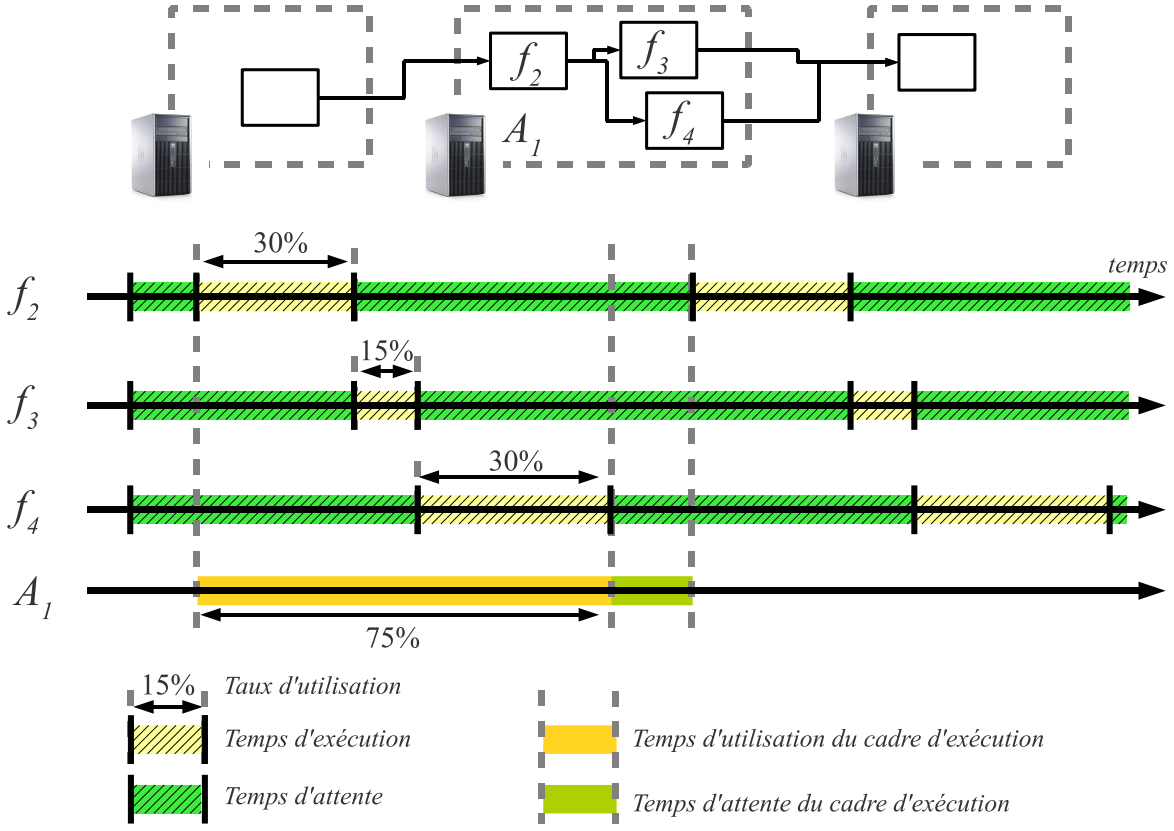


FIG. 6.25 – Taux d'utilisation d'un nœud de fusion et d'un cadre d'exécution, sur un intervalle de temps constant.

Nœud de fusion Les mesures proposées à ce niveau de l'implémentation renseignent sur le système de fusion indépendamment de la configuration déployée. Ainsi nous avons choisi trois mesures, représentées sur les Figure 6.24 et 6.19 : le temps d'attente, le temps de transmission et la taille des files d'attente.

Le temps d'attente correspond à l'intervalle de temps entre l'instant de fin de transmission d'un résultat et la fin de la préparation d'un vecteur de données à traiter. Cette mesure est utilisée par la suite pour le calcul du taux d'utilisation d'un cadre d'exécution. Le temps de transmission est calculé pour un nœud de fusion donné. Il correspond à l'intervalle de temps entre l'instant de production d'un résultat et l'instant où le dernier nœud de fusion connecté a reçu le résultat produit. En effet, la sémantique de transmission d'un résultat est basée sur l'envoi séquentiel de cette information à tous les nœuds de fusion consommateurs. Le nombre de résultats partiels stockés définit la taille des files d'attente et nous ne fixons pas de limites à la capacité de celles-ci à ce niveau de description.

Cadre d'exécution Les sondes du système produisent deux mesures pour un cadre d'exécution donné : le taux d'utilisation et la quantité de données en attente.

La mesure du taux d'utilisation d'un cadre d'exécution, illustrée sur la Figure 6.25, est basés sur le calcul de la part d'un intervalle de temps de référence, pendant laquelle tous les nœuds de fusion sont en attente. De là, le système de contrôle exploite ces mesures dans le but d'équilibrer la charge allouée au traitement sur l'ensemble du système. Ainsi, un cadre d'exécution sous-exploité est privilégié lors de la recherche d'une configuration plus performante. Cette proposition de reconfiguration est dans un premier temps traitée par les tâches *Analyse* et *Reconfiguration* puis la mise à jour du parcours de l'espace des configurations possibles est appliquée sur la tâche *Configuration*. La quantité de données en attente correspond à la somme des résultats partiels stockés dans les nœuds de fusion déployés sur le cadre considéré. Par ailleurs, notre approche ne définit aucune donnée propre à un cadre d'exécution.

Canal de communication Nous n'avons sélectionné dans notre approche que le temps de réponse d'un canal de communication. Cette mesure est utilisée lors du diagnostic d'une erreur sur un canal de communication donné. Ainsi, dans le cas où cette mesure varie de façon significative, une alerte est transmise à la tâche de reconfiguration.

La bande passante peut également être une mesure effectuée sur un canal de communication. Nous ne l'avons pas sélectionnée mais elle serait nécessaire dans le cas où le modèle d'analyse de performances traiterait le réseau de communication comme une ressource critique. Dans un tel cas, notre modèle d'analyse et les indices que nous calculons seraient étendus par l'ajout d'un mécanisme de séquentielisation du transfert et d'une modélisation plus fine de la taille des informations échangées.

6.8 Synthèse

Ce chapitre décrit le second apport majeur de cette thèse en décrivant le modèle de description et de déploiement d'un processus de fusion sur un système réparti de fusion d'informations. Il propose notamment de décrire un processus de fusion sous la forme d'un graphe de flots de données discrets, compatible avec les modèles de fusion d'informations couramment utilisés.

En plus de la gestion de l'exécution d'une configuration du système réparti, définie comme l'affectation des éléments du processus de fusion sur les ressources disponibles, notre modèle de déploiement décrit également un mécanisme de transition entre deux configurations. Ce mécanisme local permet le calcul d'un delta entre les configurations, afin d'effectuer une reconfiguration « au mieux », ne nécessitant pas l'interruption du traitement des informations.

Notre modèle de description d'un processus est ainsi pertinent au regard des modèles de fusion fréquents, et notre modèle de déploiement permet quant à lui l'adaptation de la configuration du système pour maintenir l'exécution du processus.

Chapitre 7

Analyse de la configuration

Ce chapitre du mémoire traite de l'analyse qualitative d'une configuration du système de fusion. Rappelons qu'une configuration est définie par :

- l'affectation des éléments de fusion sur les cadres d'exécution ;
- l'affectation des liens sur des chemins entre cadres d'exécution ;
- le choix d'une implémentation sur les cadres d'exécution utilisés ;

(le Chapitre 6 précise la définition d'une configuration).

La tâche de contrôle qui découle des points présentés dans ce chapitre, analyse d'une part la configuration actuellement déployée sur le système, et dans ce cas les résultats de l'analyse sont comparés à des mesures effectuées sur le système, et d'autre part les configurations candidates issues de l'étape de recherche d'affectations.

Dans la suite de ce chapitre nous présentons les indices de performance d'une configuration que nous avons identifiés, puis nous détaillons les étapes de construction du modèle d'analyse. Trois paragraphes présentent ensuite l'application des résultats d'analyse dans le but de la gestion de la performance du système de fusion et de l'équilibre de la charge sur les nœuds du système.

7.1 Choix des indices de performance

Indépendamment du modèle théorique sous-jacent, le but de la tâche de contrôle qui met en œuvre le mécanisme d'analyse répond à deux des objectifs que nous nous sommes fixés.

Débit moyen Le premier concerne la performance du système de fusion dont nous avons choisi de gérer un des aspects. En effet, la performance d'un système peut être formulée selon plusieurs critères que nous ne détaillons pas dans ce mémoire et nous proposons de résumer l'évaluation de la performance du système de fusion au temps de traitement d'un tuple de données depuis son entrée dans le système jusqu'à sa sortie. Dans un souci de clarté, ce temps de traitement sera représenté par le débit moyen du système à l'équilibre car, comme nous l'expliquons dans la suite de ce chapitre, nous situons notre approche dans un cadre probabiliste avec une analyse du système à l'équilibre. Dans le cas où le processus de fusion mis en œuvre fait intervenir plus d'un actionneur, ce qui sera vraisemblablement courant, le calcul du débit moyen sera basé sur le débit le plus faible.

Probabilité de défaillance Le second objectif de cette tâche de contrôle vise à réduire la survenue d'une défaillance du système. Le système est dit dans un état de défaillance s'il est dans l'incapacité de produire le résultat final du processus de fusion. Le modèle que nous utilisons doit donc permettre l'expression de la probabilité d'être dans un tel état à l'équilibre.

7.2 Le modèle *Generalized Stochastic Petri Net*

Les réseaux de Petri stochastiques généralisés [MGG84] (Generalized Stochastic Petri Net, GSPN) constituent le modèle le plus utilisé parmi les variétés de réseaux de Petri stochastiques, compromis entre des modèles trop simples (sans transition immédiate) et des modèles plus complexes (à distributions de sensibilisation non exponentielles négatives) difficiles à analyser.

Nous rappelons brièvement les caractéristiques de ce modèle en renvoyant le lecteur à [M.01] pour un exposé détaillé sur les réseaux de Petri et à [MBD98] pour une présentation spécifique des GSPN.

Définition 7.2.1. *Un réseau de Petri marqué (RdP) est un tuple* $N = (P, T, \mathrm{Pre}, \mathrm{Post}, M_0)$ *où* P *est l'ensemble des places,* T *l'ensemble des transitions,* Pre *la fonction d'incidence avant,* Post *la fonction d'incidence arrière et* $M_0 \in \mathbb{N}^{|P|}$*, le marquage initial de* N*.*

On représente graphiquement un RdP par des cercles pour les places, des traits pour les transitions et par des arcs reliant places et transitions : il existe un arc (de poids a) de p à t lorsque $\mathrm{Pre}(p,t) = a > 0$ (resp. un arc de t à p lorsque $\mathrm{Post}(p,t) > 0$).

Un marquage d'un RdP est un vecteur d'entiers de $\mathbb{N}^{|P|}$, chaque composante $M(p)$ représentant le nombre de jetons de la place p.

La sémantique du modèle RdP est donnée par la notion de franchissement (ou tir) des transitions. Une transition t est franchissable en M ssi $M \geq \mathrm{Pre}(.,p)$. Le franchissement (ou tir) d'une transition franchissable t en M, à partir de M produit un marquage $M' = M - \mathrm{Pre}(.,t) + \mathrm{Post}(.,t)$. Le *graphe d'accessibilité* du réseau dont les nœuds sont les marquages accessibles (obtenus par franchissements à partir du marquage initial M_0) du réseau, et où les arcs représentent les franchissements des transitions, donne exactement le comportement du RdP sous la forme d'un système de transitions étiquetées.

Les réseaux de Petri stochastiques généralisés (GSPN) sont une extension stochastique des RdP. L'idée intuitive est d'associer une durée de sensibilisation avant franchissement d'une transition franchissable. Pour rendre compte du caractère aléatoire de cette durée, elle est échantillonnée selon une distribution de probabilité sur $\mathbb{R}^+$. En raison de leurs propriétés et des bons résultats qu'elles fournissent à travers les modèles qui les emploient, ce sont les distributions exponentielles négatives (de densité $f(t) = \mu \exp(-\mu t)$ pour $t \geq 0$, de moyenne $\frac{1}{\mu}$), qui sont les plus utilisées. Il est cependant apparu qu'il fallait distinguer parmi les transitions, celles qui modélisent des activités à durée non nulle (dites temporisées) et celles (appelées immédiates) qui modélisent des choix (à caractère logique) instantanés. Ceci a conduit à définir le modèle GSPN.

Définition 7.2.2. *Un réseau de Petri stochastique généralisé (Generalized Stochastic Petri Net, GSPN) est un tuple* $N = (N', \mu, \mathrm{Prio})$ *où* N' *est un RdP marqué tel que* $T = T^i \biguplus T^t$ *avec :* T^i *ensemble des transitions immédiates;* T^t *ensemble des transitions temporisées.* $\mu(t) \in \mathbb{R}^+$ *est la moyenne de la distribution exponentielle négative associée à la transition temporisée* t*.* $\mathrm{Prio}(t) \in \mathbb{N}$ *est la priorité de la transition* t *: si* t *est temporisée,* $\mathrm{Prio}(t) = 0$*, sinon* $\mathrm{Prio}(t) > 0$*.*

Graphiquement, les transitions immédiates sont représentées par des traits et les transitions temporisées par des rectangles vides.

La dynamique des GSPN est modifiée par rapport à celle des RdP. Lorsqu'aucune transition immédiate n'est franchissable en M, c'est la transition temporisée de plus courte durée échantillonnée qui sera franchie (à la fin de sa durée échantillonnée de sensibilisation). Lorsqu'au moins une transition immédiate est franchissable en M, une transition immédiate t de l'ensemble $I(M)$ de toutes les transitions immédiates de plus grande priorité, franchissables en M est franchie avec une probabilité $\frac{\mathrm{Prio}(t)}{\sum_{t \in I(M)} \mathrm{Prio}(t)}$.

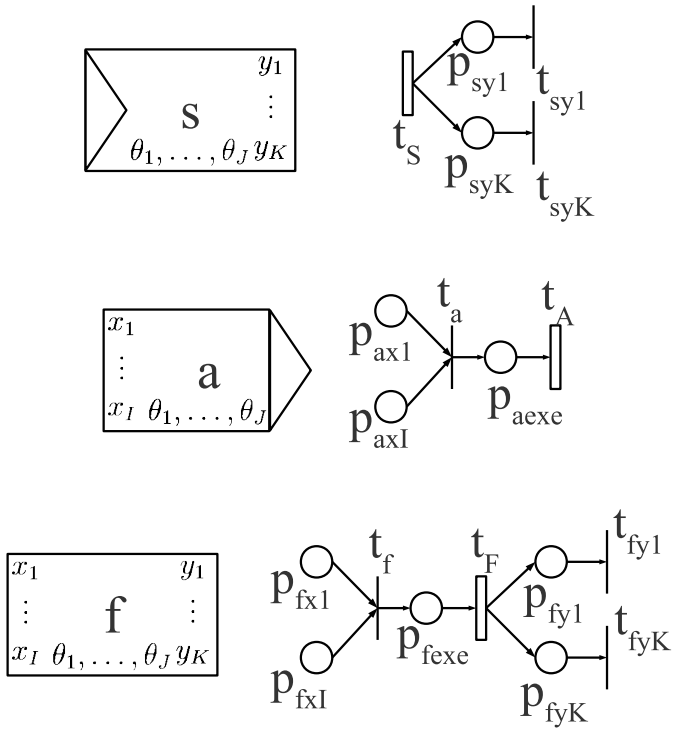


FIG. 7.1 – Traduction dans notre modèle d'analyse de l'implémentation des éléments de fusion.

Le graphe d'accessibilité d'un GSPN induit un processus semi-markovien X. Cependant, il est possible d'obtenir une chaîne de Markov Y, réduite aux marquages (dits tangibles) à durée de séjour non nulle. Les indices du processus X, à temps fini ou à l'équilibre, qui font référence à des durées (temps de séjour moyen, débit moyen des transitions temporisées, ...) peuvent alors être obtenus à partir de la chaîne Y.

7.3 Construction du modèle

Dès qu'une configuration doit être analysée, notre système en construit un modèle GSPN. Les paragraphes suivants présentent la traduction de chaque élément de la configuration.

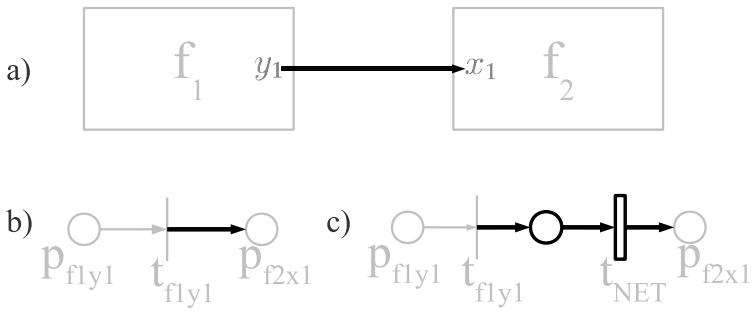


FIG. 7.2 – Traduction dans notre modèle d'analyse de l'implémentation d'un lien entre deux éléments de fusion (a) en une étape locale (b) ou distante (c).

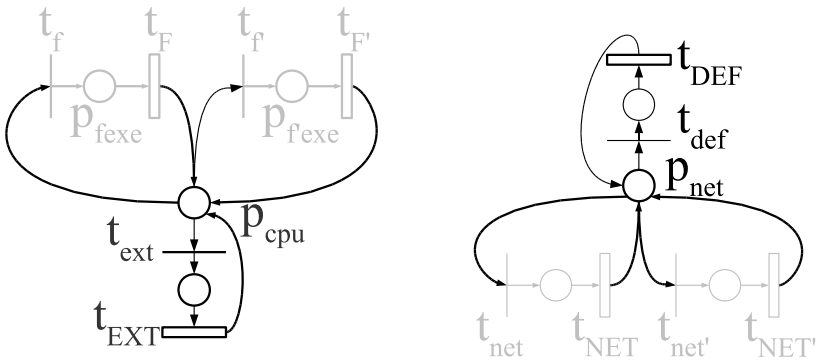


FIG. 7.3 – Traduction dans notre modèle d'analyse des ressources CPU et réseau : en grisé les éléments déjà traduits.

7.3.1 Nœud de fusion

Un nœud de fusion, illustré par la Figure 7.1, est traduit différemment selon qu'il s'agit d'une fonction de fusion, d'une source ou d'un actionneur. Prenons le cas d'une fonction de fusion f donnée, que nous traduisons depuis ses ports de donnée jusqu'à ses ports de résultat.

(1) Chaque port d'entrée x_i est traduit par une place d'entrée notée p_{fxi} reliée à une transition immédiate notée t_f commune aux places d'entrée. Cette transition représente la synchronisation des données consommées par la fonction de fusion. Par ailleurs nous ne faisons pas état ici d'un ordre d'arrivée sur les données en attente. En effet, même si nous supposons que la mise en œuvre du processus de fusion respecte l'hypothèse FIFO, la valeur des données traitées n'est pas modélisée, et donc non utilisée dans le modèle GSPN. Les jetons en attente dans les places d'entrée sont donc, conformément au modèle réseau de Petri, indifférenciés.

(2) La transition immédiate t_f est en suite reliée à une place qui modélise le démarrage du traitement. Cette place, notée p_{fexe}, est reliée à une transition temporisée t_F dont le temps de franchissement suit une une loi de probabilité exponentielle négative dont la moyenne correspond à des paramètres définis dans le Paragraphe 7.5.

(3) Le tir de la transition t_F dépose des jetons dans les places de résultat p_{fyk} associées aux ports de résultat y_k. Chaque port de résultat est finalement relié à une transition immédiate t_{fyk} dans le but de permettre la diffusion d'un résultat intermédiaire aux autres fonctions de fusion.

Une source d'information, ou un actionneur, est un cas spécial de fonction de fusion dont nous proposons des simplifications. Ainsi, une source d'information ne fait intervenir que les étapes (2) et (3) présentées précédemment et un actionneur que les étapes (1) et (2).

7.3.2 Étape locale

Un lien $l = (e_1.y, e_2.x)$ a deux traductions différentes selon qu'il fait intervenir un ou plusieurs cadres d'exécution, comme l'illustre la Figure 7.2. Dans le cas d'une étape locale les deux éléments de fusion e_1 et e_2 sont affectés au même cadre d'exécution et la transition immédiate t_{e1y} est reliée à la place d'entrée p_{e2x}.

7.3.3 Étape distante

La deuxième traduction d'un lien, également illustrée par la Figure 7.2, est utilisée lorsque le lien est implémenté par un chemin entre deux cadres d'exécution. Dans ce cas, chaque canal de communication emprunté est modélisé par une place et une transition temporisée, notée t_{NET},

dont la durée de sensibilisation correspond à un des paramètres du modèle présentés dans le Paragraphe 7.5.

7.3.4 CPU et réseau

La traduction que nous proposons n'est pas l'unique possible. En effet, selon la finesse de l'analyse que l'on souhaite effectuer, plusieurs modifications peuvent être apportées. Dans notre traduction, nous modélisons les ressources physiques utilisées comme le processeur ou une connexion réseau, illustrée par la Figure 7.3, alors que nous supposons que la taille de la mémoire disponible est infinie. Ce type d'extensions complétera les résultats de l'analyse des phénomènes d'inter-blocages entre les nœuds de fusion déployés sur un même cadre d'exécution.

Ici nous proposons d'associer à chaque cadre d'exécution une place p_{cpu} et deux transitions t_{ext} et t_{EXT} pour modéliser d'une part le déploiement de plusieurs nœuds de fusion sur le même cadre, et d'autre part le partage de la ressource avec d'autres applications indépendantes du processus de fusion. Ainsi, pour un élément de fusion donné e, la transition t_E ne sera tirée que si le processeur est disponible. Sur le même principe, nous associons à chaque canal de communication une place p_{def} et deux transitions t_{def} et t_{DEF} pour modéliser les défaillances possibles. La transition t_{NET} d'une étape distante ne sera tirée que si le canal de communication est disponible.

7.3.5 Système réparti de fusion d'informations

La suite de ce paragraphe présente deux exemples de traduction de configurations faisant intervenir respectivement un et plusieurs cadres d'exécution. L'analyse que nous souhaitons mener est à nombre d'états fini et nous souhaitons également vérifier la vitalité des réseaux que nous modélisons, notamment pour l'étude des cas de défaillance. Ainsi, la traduction d'une configuration est étendue par l'ajout de boucles entre des familles « d'actionneurs » et certaines « sources d'informations ». Nous détaillons la construction de ces boucles après les deux exemples de traduction.

7.3.6 Exemples

Un seul cadre d'exécution L'exemple que nous proposons sur la Figure 7.4 décrit une configuration qui met en œuvre un processus de fusion composé d'une source d'information, de deux fonctions de fusion et d'un actionneur. Ces quatre éléments sont tous affectés sur le même cadre d'exécution ce qui implique l'utilisation de la première traduction de l'implémentation d'un lien. En outre, cet exemple illustre l'indépendance de l'analyse de la configuration, vis-à-vis des ports de paramètres et le port $f_1.\theta_1$ n'est donc pas traduit dans le modèle d'analyse.

Plusieurs cadres d'exécution La Figure 7.5 illustre le même processus de fusion que l'exemple précédent, mais plusieurs cadres d'exécution sont utilisés. Ces cadres sont ici installés sur différents PC dans un souci de clarté. Le réseau supportant le transfert des résultats relie les couples *PC1* à *PC2*, *PC2* à *PC3* et *PC3* à *PC4*. Chacun des éléments de fusion du processus est affecté à un cadre d'exécution différent ce qui implique que les liens entre ces éléments sont implémentés par des chemins dans le réseau. Ces chemins sont composés d'étapes traduites par des transitions temporisées entre les places qui modélisent les ports d'entrée et de résultat des éléments de fusion.

On notera que tous les réseaux de Petri obtenus sont des *graphes marqués* : chaque place a exactement un arc entrant et un arc sortant. Il n'y a donc pas de conflit dans nos systèmes mais uniquement des synchronisations.

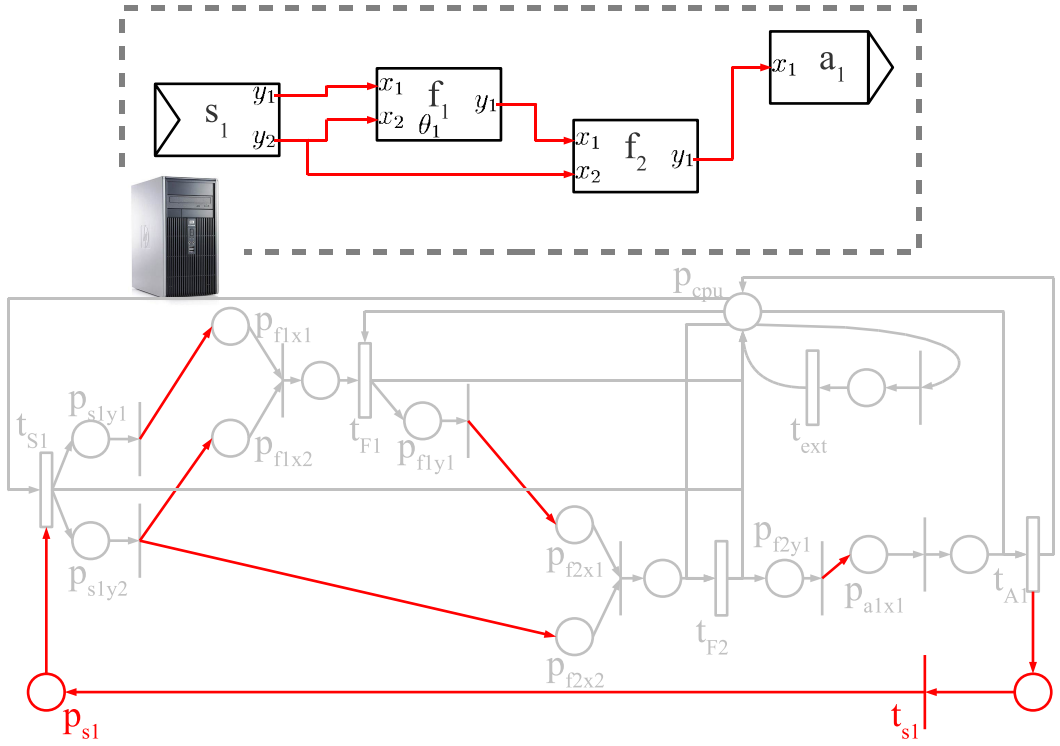


FIG. 7.4 – Traduction dans notre modèle d'analyse de l'implémentation d'un processus de fusion sur un cadre d'exécution : en grisé les éléments déjà traduits.

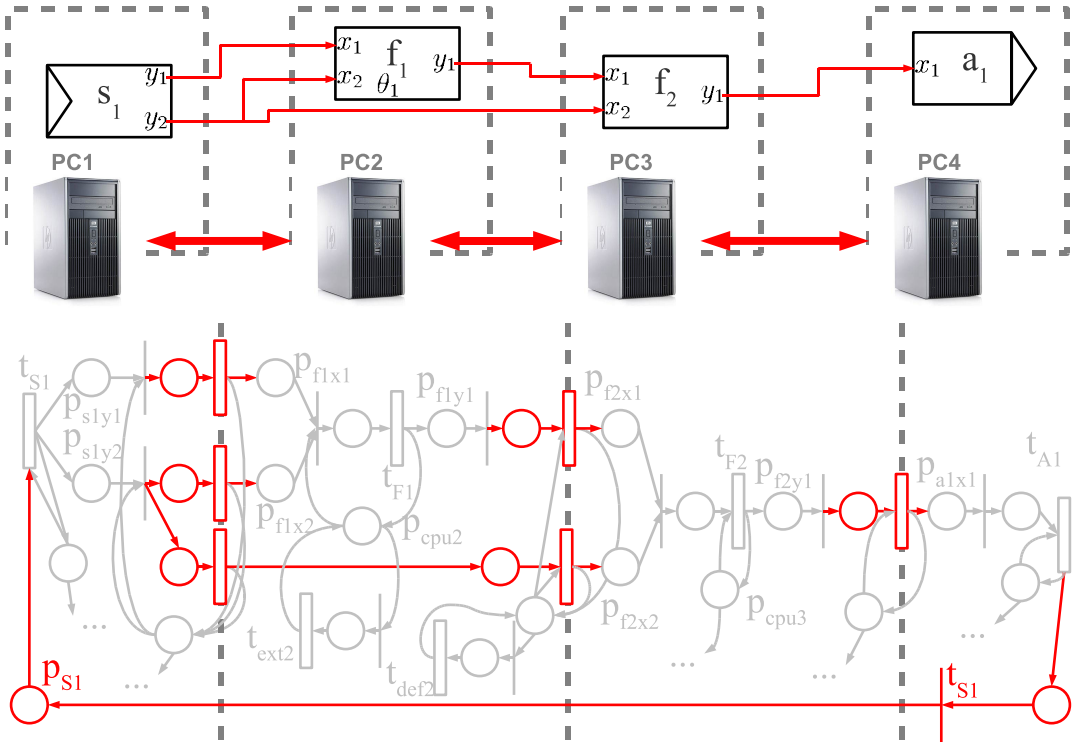


FIG. 7.5 – Traduction dans notre modèle d'analyse de l'implémentation d'un processus de fusion sur plusieurs cadres d'exécution : en grisé les éléments déjà traduits.

7.4 Modèle fini

La finitude du graphe d'accessibilité du réseau de Petri est indispensable pour pouvoir calculer des indices de performances du système. Elle correspond à la finitude de la chaîne de Markov à temps continu sous-jacente. Dans ce but, et de manière classique, on ajoute des places marquées et des transitions au modèle garantissant cette finitude. On parle alors de « bouclage du modèle d'analyse ».

7.4.1 Description

Le modèle du système définit au moins une boucle d'analyse, telle que celle illustrée dans les Figures 7.4 et 7.5 et il y a exactement autant de boucles d'analyse que de sources s de données et ceci pour toute configuration du système. Une partie de cette boucle est constituée par une place p_s reliée à la transition temporisée de la source considérée, comme l'illustre la Figure 7.6. L'autre partie de la boucle est constituée d'autant de places p_{as} qu'il y a d'actionneurs qui consomment un résultat basé sur l'information produite par s. Toutes les places qui appartiennent à la même boucle d'analyse sont reliées à une transition immédiate elle-même reliée à la place p_s définie pour chaque boucle.

On vérifie aisément que chaque boucle définit un ensemble de P-semiflots, c'est à dire un ensemble de places $\{p_1, \ldots, p_m\}$ pondérées par des entiers positifs $\{a_1, \ldots, a_m\}$ tels que $\sum_{i=1}^{m} a_i M(p_i) = C$ pour tout marquage accessible M. La notion de P-semiflot rend compte d'une propriété de conservation d'entités dans les réseaux de Petri, et l'ensemble des P-semiflots correspondant à une source s et ses actionneurs associés (qui consomment des données que s a produites) témoignent de la conservation des données issues de s jusqu'à la production des résultats auxquels elles contribuent.

À chaque semi-flot F ci-dessus, du fait que le GSPN est un graphe marqué, nous pouvons associer la suite de transitions temporisées qui ont une place de F en entrée. Ces transitions constituent un « chemin » à travers le système, qu'empruntent des « entités virtuelles » générées par les données de la source s. Ainsi, à l'équilibre stochastique, le débit moyen à travers toutes les transitions d'un chemin doit être identique, égal à celui des entités sur ce chemin.

7.4.2 Construction

La construction des boucles d'analyse dans le modèle du système peut être réalisée de plusieurs façons, à partir du réseau de Petri final, ou à partir du processus de fusion. Nous avons choisi de baser la nôtre sur la description du processus de fusion présentée dans la Chapitre 6 car cette description contient exactement les informations nécessaires. Ainsi, pour un processus de fusion donné $\mathcal{P} = (\mathcal{F}, L)$, l'ensemble des éléments de fusion est composé des trois sous-ensembles disjoints, respectivement pour les sources d'informations $\mathcal{S}$, pour les fonctions de fusion et pour les actionneurs $\mathcal{A}$.

Si l'on considère l'ordre partiel d'exécution $(\mathcal{P}, <)$ défini dans le Chapitre 6, pour chaque source d'information $s \in \mathcal{S}$, une boucle d'analyse est définie entre s et l'ensemble d'actionneurs $A \subseteq \mathcal{A}$ tel que $a \in A \Leftrightarrow s < a$.

7.5 Paramètres du modèle

Si la structure du modèle d'analyse dépend de la configuration du système, certain paramètres doivent le compléter pour permettre le calcul du débit moyen du système ainsi que la probabilité d'être dans un état de défaillance. Nous avons identifié les trois familles de paramètres suivantes :

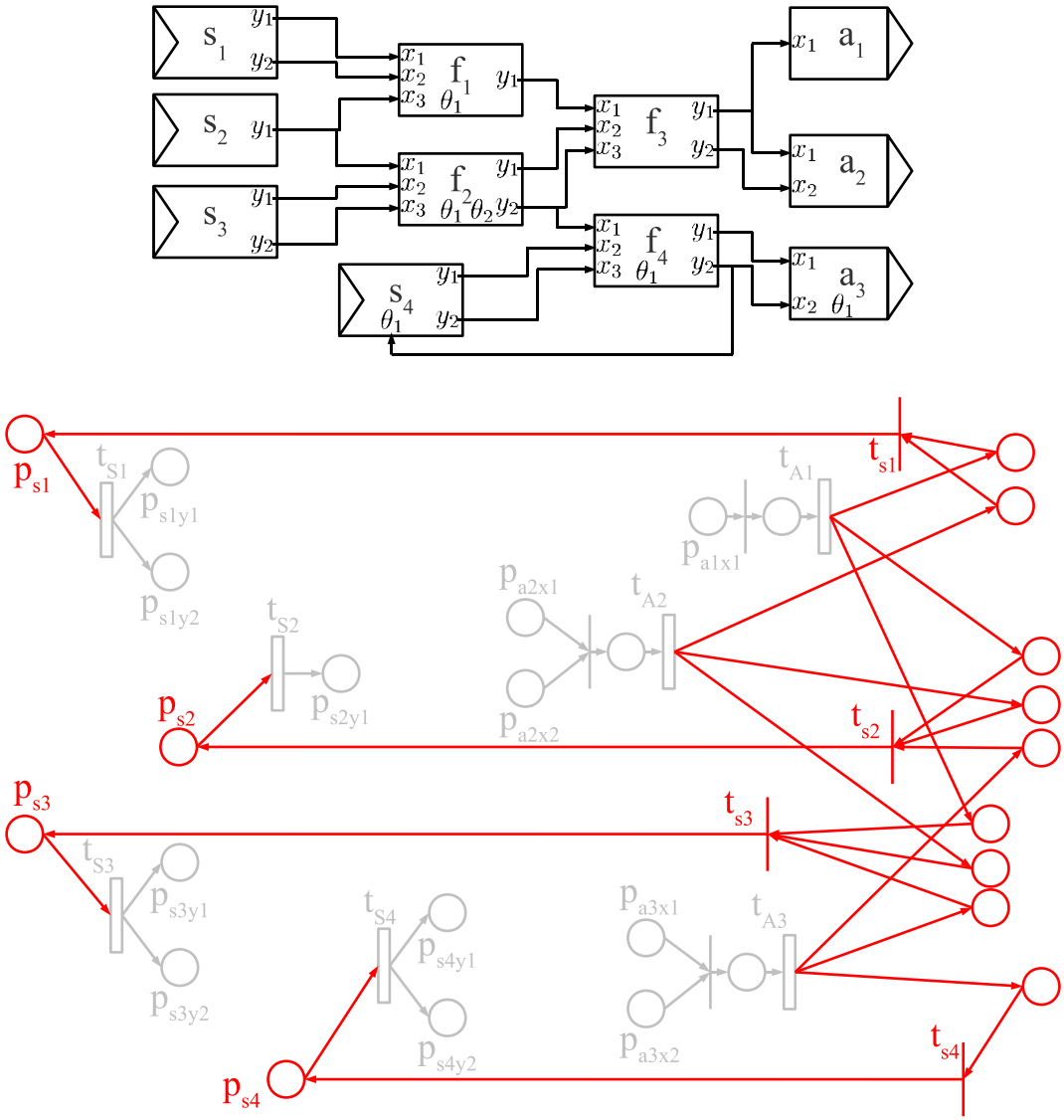


FIG. 7.6 – Description d'un processus de fusion puis bouclage d'analyse associé : en grisé les éléments déjà traduits.

- les temps moyens de traitement, traduits par les temps de franchissement des transitions t_E avec e un élément de fusion et donc la moyenne de leur distribution ;
- les temps moyens de transfert, traduits par les temps de franchissement des transitions t_{NET} qui modélisent des communications dans le réseau, et donc aussi la moyenne de leur distribution ;
- le nombre de « vagues » de données en traitement, modélisé par le nombre de jetons dans le marquage initial du modèle.

Temps de traitement La configuration du système définit une implémentation pour chaque affectation d'un élément de fusion sur un cadre d'exécution et un temps de traitement est associé à chaque transition temporisée associée à l'élément considéré. Le temps de sensibilisation μ_t de ce type de transition est supposé disponible lors de la construction du modèle du système. En effet, ce paramètre du modèle peut être fourni par des spécifications techniques du matériel, ou directement estimé grâce aux mesures effectuées sur le système (c.f. le Paragraphe 6.7.2). Afin de réaliser un premier modèle d'analyse fonctionnel dans le cadre de cette thèse, nous supposons que le temps de traitement ne varie qu'en fonction de l'implémentation choisie et reste indépendant de la donnée présentée en entrée. Cette simplification se justifie intuitivement par l'équivalence de l'évolution des temps de traitement de deux implémentations différentes sur le même jeu de données. Ainsi, si une donnée d'entrée augmente significativement le temps de traitement d'une implémentation, nous posons qu'il en sera de même pour une autre implémentation.

Temps de transfert En suivant le même raisonnement que pour les temps de traitement, le temps de franchissement des transitions temporisées t_{net} est également paramétré. Le temps de sensibilisation de ces transitions est aussi supposé disponible lors de la construction du modèle comme nous l'avons présenté précédemment. Dans un premier temps, nous fixons l'évolution de la valeur de ce paramètre uniquement sur le temps de latence d'un canal de communication entre deux cadres d'exécution. En effet, une évaluation de la bande passante n'est pour l'heure pas exploitable pour deux raisons. Premièrement par l'indépendance entre notre modèle et la taille des données échangées et deuxièmement par la simplification que nous avons apportée sur l'utilisation partagée des canaux de communication.

Nombre de vagues Lors de la définition des boucles d'analyse, dans le Paragraphe 7.4, le marquage initial des places associées aux sources d'information correspond au nombre maximal de « vagues » de résultats intermédiaires en cours de traitement. Pour simplifier l'analyse, nous proposons de fixer le même nombre de jetons dans les places P_{si} et de définir ainsi le dernier paramètre du modèle. Dans un premier temps nous supposons également que les autres places du modèle sont initialement vides ; cette hypothèse doit être revue si la traduction d'une configuration était étendue (places de ressources).

7.6 Calcul des indices

Le calcul des indices de performances que nous souhaitons analyser s'effectue à l'équilibre avec un modèle et des paramètres associés constants. Nous avons sélectionné trois indices que le mécanisme de calcul doit fournir.

Le **nombre moyen de jeton en attente** pour un élément de fusion donné ou pour un cadre d'exécution donné. Cet indice nous permet d'évaluer la taille des files d'attente implantées dans les nœuds de fusion, dans une approche prédictive, mais également de fixer des plafonds au-delà desquels une alarme est émise. Si la valeur de cette indice est « proche » du nombre de vagues

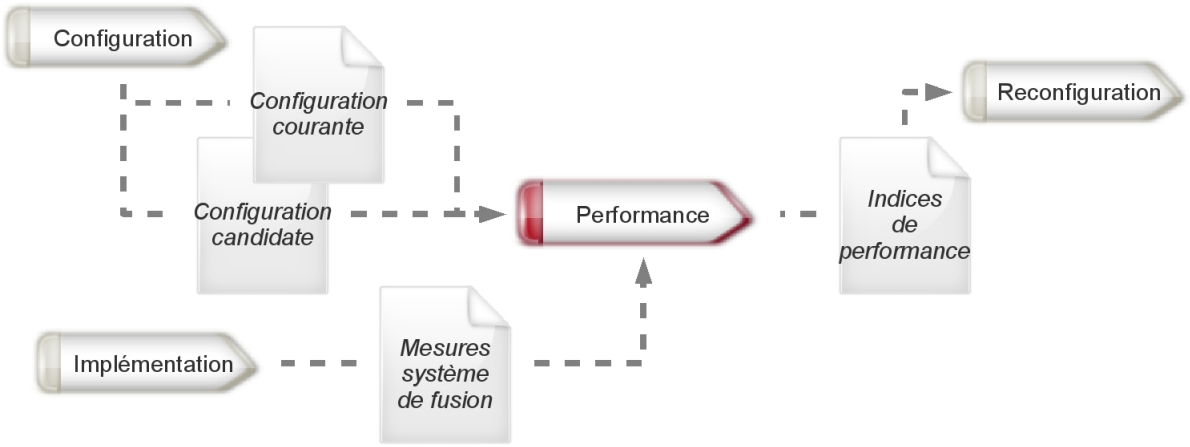


FIG. 7.7 – Tâche de contrôle liée à la performance.

de traitement fixé en paramètre, alors la configuration sera écartée car elle présente des risques de de phénomènes d'engorgement.

Le **temps moyen de traversée des jetons** est un indice qui nous permet d'évaluer la puissance de la configuration analysée. Ce temps correspond au « temps de réponse moyen » de la traversée des entités virtuelles depuis le franchissement d'une transition temporisée associée à une source d'information, jusqu'à leur consommation par une transition associée à un actionneur. La valeur de cet indice sert par la suite de critère de sélection entre deux configurations candidates.

Le dernier indice calculé concerne **l'intervalle de temps moyen d'inter-production de deux résultats finaux**, ou plus formellement le débit moyen du système mesuré à chaque actionneur. Cet indice renseigne sur la réactivité du système au même titre que le premier indice que nous proposons. En effet, si un phénomène d'engorgement est détecté suite à l'analyse du nombre de jetons en attente, cet engorgement sera confirmé par un intervalle de temps élevé pour les actionneurs consommant des données concernées par l'engorgement. En outre, cet indice reflète également l'équilibre des charges de travail affectées à chaque cadre d'exécution.

Ces indices sont évalués pour une configuration donnée, courante ou une candidate, et pour un jeu de paramètres fixé. Par ailleurs, plusieurs calculs d'indices seront exécutés en même temps sur plusieurs machine, avec une résolution centralisée de chaque calcul. Le parallélisme est alors exploité sur différents jeux de paramètres.

7.7 Gestion de la performance

L'analyse de la performance d'une configuration s'inscrit dans le système de contrôle selon plusieurs interactions que nous détaillons dans les paragraphes ci-dessous.

7.7.1 Création d'un modèle

La première interaction a lieu avec la tâche de contrôle en charge du parcours de l'espace des configurations possibles. Deux types de configuration peuvent être analysés, selon qu'il s'agisse d'un ordre de déploiement ou d'analyse. Dans le premier cas, la configuration à déployer est transmise aux tâches « Performance » et « Implémentation ». Dans le second cas, la configuration n'est transmise qu'à la tâche « Performance ». Dès la réception d'une configuration, le système construit le modèle d'analyse en suivant les traductions élémentaires que nous avons exposées

dans le Paragraphe 7.3. Seule la structure du modèle d'analyse résulte de cette interaction entre les tâches « Configuration » et « Performance » alors que la mise à jour des paramètres du modèle provient d'une seconde tâche de contrôle.

7.7.2 Mise à jour des attributs

La tâche de contrôle « Performance » réagit également à des messages envoyés par la tâche « Implémentation ». Dès qu'une configuration est déployée par cette tâche, les éléments de contrôle inclus dans la mise en œuvre du processus de fusion émettent l'ensemble de mesures que nous avons présenté dans le Paragraphe 6.7.2. Ces mesures sont utilisées dans un premier temps pour paramétrer les modèles d'analyse des configurations évaluées (rappelons que plusieurs configurations sont évaluées en parallèle) et dans un deuxième temps elles sont comparées aux indices de performance calculés à partir du modèle de la configuration courante. En effet, la configuration courante est au plus tard analysée lors de son déploiement - car lors des premiers déploiements le système dispose de peu de détails sur le matériel. Les mesures produites par la tâche « Implémentation » sont ainsi comparées aux indices de performance calculés et un message d'alarme est éventuellement généré si le comportement du système est significativement différent de celui prévu. L'étude du seuil d'émission d'un message d'alarme sort du cadre de cette thèse et les détails techniques de cette tâche de contrôle seront fixés suite à une série d'expérimentations.

Les messages d'alarme et les indices de performance associés aux configurations candidates sont finalement envoyés à la tâche « Reconfiguration » qui prendra la décision de modifier ou non la configuration courante.

7.8 Simplification de la recherche d'une configuration

Objectif Indépendamment du mécanisme d'analyse d'une configuration, nous proposons des simplifications du parcours de l'espace des solutions, effectué dans la tâche « Configuration ». Ces simplifications sont basées sur des résultats préliminaires et nous permettent d'écarter des configurations peu performantes avant leur analyse.

Les propositions que nous présentons dans la suite de ce paragraphe constituent des pistes que nous envisageons d'explorer lors d'expérimentations à venir.

Attributs et mesures de la configuration Dans la présentation du modèle de description du Chapitre 6, nous avons introduit plusieurs mesures possibles d'une configuration (Paragraphe 6.4.2).

La première concerne le nombre d'étapes nécessaires à l'implémentation d'un lien. Une des hypothèses, que les expérimentations devraient vérifier, énonce que la probabilité du système d'être dans un état de défaillance croît avec le nombre d'étapes d'un chemin.

Le nombre de cadres d'exécution utilisés pour déployer une configuration est notre seconde piste de simplification. Il est en effet raisonnable de penser que le temps de traitement diminue avec l'augmentation de la répartition des fonctions de fusion sur le système.

Orientation de la recherche de configuration Nous proposons, dans un premier temps, de nous limiter aux deux mesures précédentes pour orienter le parcours de l'espace des configurations possibles. Nous avons présenté un algorithme de recherche par contrainte tenant compte de ces pistes dans [BHMP09]. Cet algorithme, non encore implémenté, sera basé sur l'optimisation d'une fonction de coût.

7.9 Synthèse

En conclusion de ce chapitre, rappelons que la partie du système de contrôle en charge de l'analyse des performances du SRFI joue un rôle important dans la sélection de la configuration du système. Le mécanisme de traduction que nous proposons, même s'il n'est pas totalement automatisé, est fonctionnel à la vue des premiers résultats issus des travaux en cours. L'étude de la complexité de notre algorithme, notamment l'expression du nombre de marquages du modèle, est également en cours et donnera lieu à communication.

En parallèle de cette étude de l'algorithme de traduction, nous travaillons également sur l'exploitation des caractéristiques structurelles du modèle. Ainsi, alors que nous évaluons plusieurs configurations candidates sur autant de cadres d'exécution, nous pensons répartir l'évaluation d'une configuration sur un ensemble de cadres. La structure du modèle, dérivée de celle de la configuration, serait alors à la base d'évaluations focalisées sur des parties du système.

Chapitre 8

Surveillance et reconfiguration, propositions

Ce chapitre du mémoire concerne les deux tâches de contrôle pour lesquelles nous proposons des pistes de solutions. Le Paragraphe 8.1 reprend les problématiques abordées lors de la détection des erreurs et des défaillances du système de fusion et nous avançons ensuite deux pistes que nous évaluons vis-à-vis des priorités de notre système de contrôle.

La seconde partie de ce chapitre regroupe le dernier élément du système de contrôle. Celui-ci a pour but la prise de décision concernant le déclenchement d'une reconfiguration du système. Là encore nous proposons un axe de travail du domaine des perspectives.

8.1 Détection d'erreurs et de défaillances

À présent qu'un processus de fusion est décrit (Chapitre 6), et qu'une implémentation est déployée et analysée (Chapitre 6 et 7), nous pouvons traiter de la détection d'une défaillance du système.

Pour cela, nous nous sommes intéressés à une méthode qui s'inscrit dans le processus d'extraction de connaissance dans les données (ECD) et qui construit un modèle de prédiction de défaillances. Certains détails nécessitent l'analyse du système final en cours d'exécution, ce qui est pour l'heure impossible. La démarche globale reste néanmoins valide et les points abordés sont de bonnes pistes pour les travaux de recherche à venir.

Dans la première partie de ce paragraphe, nous posons les définitions d'une erreur et d'une défaillance, nécessaires à la conception de notre modèle de prédiction. Le Paragraphe 8.1.2 présente plusieurs approches de diagnostic et de pronostic de défaillances dont celle que nous avons retenue.

Enfin, le Paragraphe 8.1.3 traite de la tâche de contrôle qui met en œuvre un processus de fouille de données dans le cadre de notre système de contrôle.

8.1.1 Erreurs et défaillances

Notre approche définit plusieurs axes d'erreurs et de défaillances dans le système. En effet, conformément à [ALRL04, Lap95], nous adoptons la définition d'une défaillance comme étant la conséquence d'une erreur, elle-même découlant d'une faute.

Le système que nous construisons propose un mécanisme de détection d'erreurs et de défaillances provoquées par des fautes issues d'éléments extérieurs au système, comme par exemple des ressources physiques ou des implémentations de fonctions de fusion fournies par un développeur. La détection et la correction des fautes produites par les composants du système d'exécution et

du système de contrôle sont par ailleurs effectuées pendant la phase de développement de notre outil.

Certaines erreurs dues au système de contrôle seront néanmoins détectées puis corrigées. Celles-ci découlent de l'imperfection inhérente à tout système informatique. Nous avons donc choisi de concentrer nos efforts sur la création d'un système adaptatif plutôt que sur celle d'un système exempt de tout défaut.

Dans la suite de ce paragraphe, nous détaillons quelles sont les erreurs et les défaillances que nous détectons. Celles-ci peuvent se situer au niveau des cadres d'exécution ou des nœuds de fusion, comme nous les présentons selon deux axes dans [BHMP09].

Erreur

Définition 8.1.1 (Erreur)**.** *Une erreur découle d'un comportement fautif. Elle appartient à une des catégories suivantes :*

1. *les erreurs de transmission de données entre deux cadres d'exécution ;*
2. *les erreurs d'exécution d'une entité d'exécution ;*
3. *les erreurs du système d'exécution induite par le système de contrôle.*

Erreurs de transmission de données

Les erreurs de transmission de données surviennent lors de la perte partielle d'un canal de communication. En effet, nous supposons qu'un canal est fiable, i.e. qu'il n'altère pas les données transmises, mais dont la disponibilité est variable.

Détection Les erreurs de transmission de données sont détectées par les nœuds de fusion déployés dans le système. Ainsi, le nœud de fusion à l'initiative du transfert détecte un éventuel problème sur le canal de communication. Si la perte totale du canal n'est qu'un cas extrême d'erreur, notre approche identifie plutôt une erreur de transmission de données comme la perte temporaire du canal, ou la modification significative de la bande passante et de la vitesse de transmission.

Erreurs d'exécution

Les erreurs d'exécution que nous gérons sont issues d'une faute générée par l'implémentation d'une entité d'exécution. De telles erreurs sont légion dès lors qu'un développeur est peu rigoureux : les exemples les plus courants sont le dépassement d'espace mémoire ou la mauvaise gestion d'une connexion à un tiers service.

Détection Un seul nœud de fusion est nécessaire à la détection d'une erreur d'exécution. En effet, celle-ci survient quand l'entité d'exécution génère une erreur lors de son exécution. Si celle-ci n'est pas « capturée » dans le code de l'implémentation, elle l'est finalement par le mécanisme de contrôle présent dans le nœud de fusion. De là, le système de contrôle est informé de cette erreur et une reconfiguration peut être demandée.

Erreurs de contrôle

Les erreurs de la dernière catégorie sont introduites par le système de contrôle. Elles découlent du déploiement d'une configuration mal évaluée qui induit un des états suivants :

- un inter-blocage sur une ressource ;

– ou une saturation d'une partie du système.

Une des causes du premier état repose sur une incompatibilité des entités d'exécution, due dans certains cas à des ressources partagées au sein d'un cadre d'exécution : comme par exemple une connexion série. Une saturation peut au contraire être introduite par le système de contrôle. En effet, suite à une mauvaise évaluation de la configuration du système, l'affectation des nœuds de fusion peut être déséquilibrée. Ceci implique dans certains cas une saturation des zones de stockage du système.

Détection Une erreur de cette catégorie est liée à une action inappropriée du système de contrôle. Une telle erreur est détectée quand une des files d'attente d'un nœud de fusion contient un nombre anormalement élevé de données, selon un plafond établi lors de l'analyse de la configuration. Toujours en nous basant sur l'analyse de la configuration, un temps d'exécution « normal » peut être approximé et tout écart génère alors une erreur au niveau du nœud d'exécution.

Défaillance

Définition 8.1.2 (Défaillance). *Nous définissons deux défaillances du système :*
- *une défaillance du système d'exécution*
- *une défaillance du système de contrôle.*

Le système que nous proposons repose sur un duo de sous-systèmes, respectivement pour l'exécution d'un processus de fusion et pour son contrôle. De ce fait, chaque sous-système peut potentiellement être défaillant ce qui implique, dans notre cas, que le système global est défaillant.

La première défaillance concerne le système d'exécution, dans lequel une erreur (Définition 8.1.1) n'a pas été corrigée. Par conséquent, un ou plusieurs cadres d'exécution sont dans l'incapacité de produire un résultat alors que des données sont en attentes de traitement.

Une défaillance du système de contrôle a lieu quand un cadre d'exécution n'assume plus une des tâches de contrôle qui lui est attribuée. Par ailleurs, les deux défaillances du système sont indépendantes et il est possible qu'un des deux sous-systèmes soit défaillant sans pour autant que l'autre ne le devienne.

8.1.2 Prévention

Plusieurs approches sont possibles pour prévenir la survenue d'une erreur et *a fortiori* d'une défaillance. La première d'entre elles, dite préventive, réside dans l'adoption de méthodes formelles de conception, tant au niveau de la description et de la preuve des algorithmes implémentés, qu'au niveau du développement de l'implémentation du système. D'autres approches proposent au contraire un diagnostic du système lorsqu'une erreur ou une défaillance est détectée avec dans certains cas un pronostic au *runtime* de la défaillance ou de l'erreur avant son occurrence.

Réseaux bayésiens

[Kle92] propose les bases du diagnostic de défaillance basé sur les réseaux bayésiens et, [PWN09] a attiré notre attention lors de nos investigations. L'approche présentée est basée sur des réseaux bayésiens et peut être appliquée à notre système. Mais comme d'autres types de modèles d'analyse, il est nécessaire d'une part de construire un modèle du système et d'autre part d'appliquer des calculs sur ce modèle afin d'obtenir les caractéristiques du système.

Ce type d'analyse implique donc que les informations nécessaires à la création du modèle du système sont disponibles. Hors, même si nous pouvons déduire la structure du modèle depuis la configuration du système (nous l'avons par ailleurs réalisé dans le Chapitre 7) les valeurs des attributs de ce modèle sont par nature inconnues. En effet, si une erreur survient, c'est avant

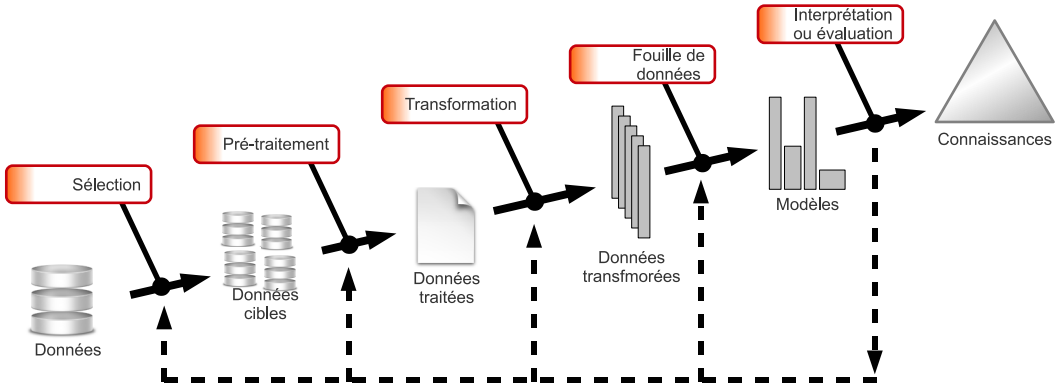


FIG. 8.1 – Le processus d'extraction de connaissance dans les données.

tout parce qu'elle n'a pas été détectée par l'analyse formelle préliminaire. En outre, le temps d'exécution nécessaire à l'analyse du modèle du système peut, souvent, être supérieur à l'horizon temporel autorisé avant l'occurrence de la défaillance.

Pour ces raisons, nous avons sélectionné une autre piste qui propose un diagnostic de défaillance pendant l'exécution.

Extraction de connaissance et fouille de données

Notre approche est basée sur le processus *ECD* et l'algorithme de fouille de données *WinMiner* [MR04] tels qu'ils ont été utilisés par une autre équipe de recherche de notre laboratoire. Cette méthode implique la constitution puis la sélection d'un historique de données représentatives du comportement de notre système. Ces données sont ensuite codées et transformées en une séquence d'évènements, comme l'illustre la Figure 8.2.

Enfin, l'algorithme de fouille de données *WinMiner* est appliqué à ces séquences d'événements et produit en résultat une base comportementale de notre système sous la forme de « règles d'épisodes ». Pour cela, *WinMiner* génère un ensemble « d'épisodes série » composé des motifs fréquents, ordonnés temporellement, pour une séquence d'événements donnée.

Dans l'exemple de la Figure 8.2, l'épisode $\alpha = A \rightarrow B$ se traduit par « l'événement A est suivi de l'événement B ». Pour une contrainte de distance entre les occurrences fixée, les occurrences de α dans la séquence S sont $occ(\alpha, S) = \{[10, 12], [11, 12], [16, 18]\}$.

Ensuite, l'algorithme présenté construit des règles d'épisodes par association des épisodes fréquents précédemment générés. Ainsi, pour l'exemple précédent, les épisodes $\alpha = A \rightarrow B$ et $\beta = A \rightarrow B \rightarrow C$ permettent la création de la règle d'épisode correspondante. Celle-ci est notée $A \rightarrow B \Rightarrow C$ telle que $\Rightarrow$ est l'opérateur logique d'implication.

Le mécanisme qui produit ces règles d'implications est soumis à une contrainte de sélection basée sur une probabilité conditionnelle. Celle-ci est calculée, pour une règle donnée et une fenêtre temporelle d'analyse variable, comme étant la probabilité que la conclusion apparaisse sachant que sa prémisse est apparue dans la séquence d'événements.

L'algorithme ne retient finalement que les règles ayant une probabilité conditionnelle maximale pour une taille de fenêtre donnée, dite « optimale » selon les critères présentés par les auteurs. [LBMV08] présente une application de ces règles pour le diagnostic et le pronostic de défaillances.

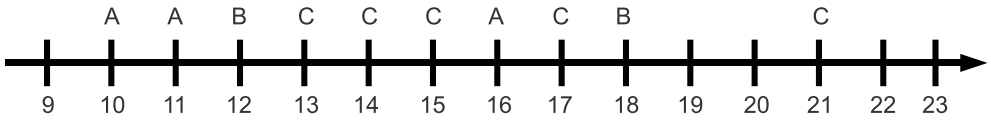


FIG. 8.2 – Exemple d'une séquence d'événements tiré de [MR04].

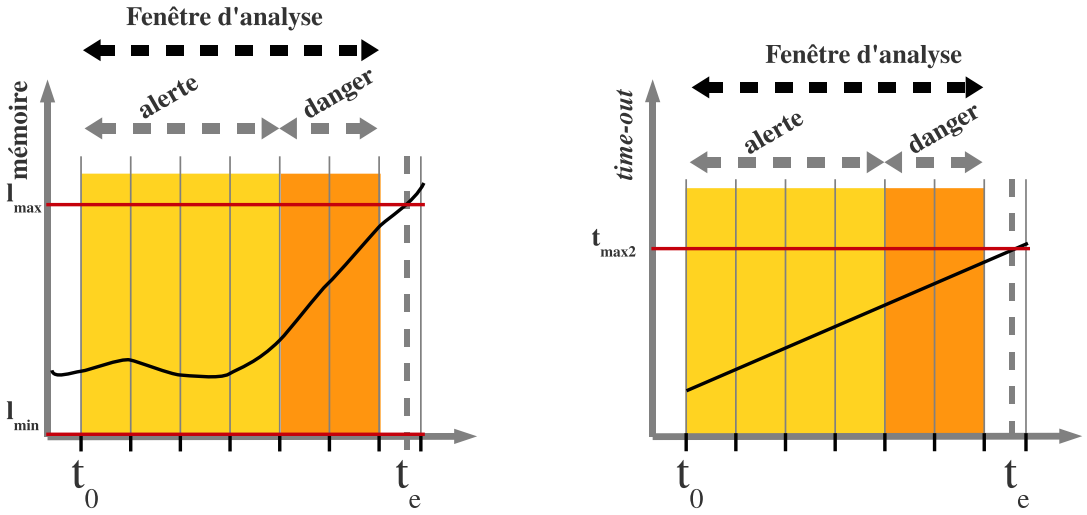


FIG. 8.3 – Analyse des mesures du système pour le diagnostic de défaillance.

Concernant les SRFI

Dans le contexte des systèmes de fusion d'informations, l'application de cette méthode n'est pas aisée. En effet, elle implique la constitution d'un historique de mesures de taille potentiellement infinie si l'on considère le temps d'utilisation du système, ou à partir de mesures issues d'historiques courts si l'on considère les périodes de fonctionnement entre les reconfigurations du système.

Une autre difficulté réside dans la sélection des mesures pertinentes. Rappelons que le nombre de sondes est potentiellement élevé (Paragraphe 6.7.2), et des algorithmes de réduction de dimension seront alors nécessaires. Enfin, la transformation de mesures continues (issues de nos sondes) en un ensemble d'événements discrets exploitable par l'algorithme de fouille de données pourrait requérir l'application d'une méthode de *clustering*. Ces derniers point représentent des domaines de recherche trop éloignés du notre et sortent du cadre de nos travaux.

Exemple

Les Figures 8.2 et 8.3 présentent les deux vues d'un exemple de prévention de défaillance. Dans la Figure 8.2, les événements notés A, B et C, correspondent à des franchissement de seuils par des mesures issues du système. Le principe de notre approche est basé sur la mise à jour de quatre ensembles d'événements :

- les défaillances ou les erreurs, identifiées *a posteriori* ;
- les événements dangereux, calculés à partir de l'analyse de l'historique de mesures, et exhibés comme les prémices d'implications qui conduisent à une défaillance ou à une erreur ;
- les événements suspects, qui conduisent dans certains cas à une défaillance ou à une erreur ;

– et les événements non identifiés.

Cette distinction entre les événements suspects et dangereux est illustrée par la Figure 8.3. Les événements suspects, représentés dans la zone « alerte », n'aboutissent pas toujours à une défaillance (celle-ci peut par exemple être évitée). Dans le cas d'un évitement, l'indice de confiance de la règle d'implication prend toute son importance car, sans lui, le système pourrait être en perpétuelle reconfiguration car tout événement peut potentiellement conduire à une défaillance.

8.1.3 Gestion de la robustesse

La tâche *Robustesse* est une des tâches de notre système de contrôle. Elle a pour but la gestion de la volatilité des ressources, ce qui est un des objectifs que nous avons formulés dans le Chapitre 2.2.2. La suite de ce paragraphe présente les trois interactions possibles entre la tâche *Robustesse* et le reste du système de contrôle, que nous illustrons dans la Figure 8.4.

Mise en place des éléments de surveillance

Les mesures que nous présentons dans le Paragraphe 6.7.2 sont fournies par un ensemble de sondes automatiquement installées dans le système. Ces sondes sont présentes dans deux éléments du système : dans les cadres d'exécution et dans les nœuds de fusion. La liste de ces sondes est donnée par la configuration du système, transmise par la tâche *Configuration* lors du déploiement.

Lecture des capteurs internes

Dès qu'une nouvelle configuration est déployée, la tâche *Robustesse* effectue une lecture périodique des sondes installées dans le système. Si la liste des sondes est fournie par la tâche *Configuration*, la prise de mesure s'effectue par la tâche *Implémentation*.

Les mesures fournies par les sondes d'un nœud de fusion sont surveillées par le cadre d'exécution qui le déploie, alors que les mesures issues d'un cadre d'exécution sont analysées par un autre cadre d'exécution. Cette analyse des mesures propose en résultat un diagnostic du système avec, le cas échéant, l'identification d'un élément potentiellement défaillant.

Demande de reconfiguration

Le diagnostic fournit suite à l'analyse des mesures produites par le système identifie, dans certains cas, un élément potentiellement défaillant. L'analyse faite par la tâche *Robustesse* aboutit alors à la transmission d'un message d'alerte à la tâche *Reconfiguration*. Ce message contient l'élément identifié comme défaillant avec un indice de confiance de l'occurrence d'une erreur sur un intervalle de temps.

En effet, si la défaillance n'est pas avérée, la tâche *Reconfiguration* est en mesure de pouvoir évaluer le gain potentiel d'une reconfiguration du système alors que celui-ci traite des informations.

8.2 Reconfiguration

La reconfiguration du système intervient pour deux raisons. Premièrement, le système de contrôle peut modifier la configuration du système si un message d'alerte est envoyé par la tâche de contrôle *Robustesse*. Celle-ci fournit alors un pronostic de panne suivant lequel la prise de décision est effectuée.

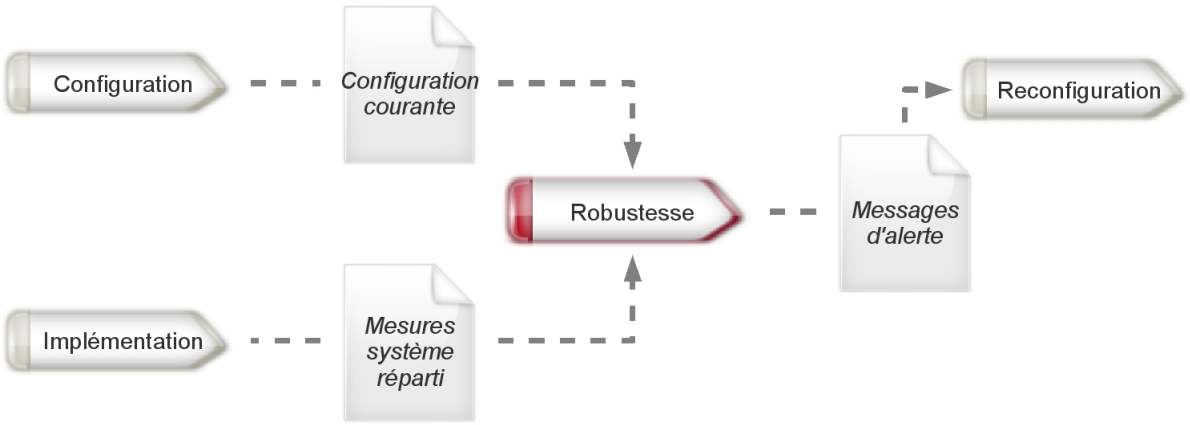


FIG. 8.4 – Tâche de contrôle liée à la robustesse.

Le besoin de reconfiguration peut également être motivé par la découverte d'une configuration plus performante que celle actuellement déployée par le système. La prise de décision est alors basée sur le potentiel gain de performance suite à la reconfiguration du système. La problématique de prise de décision peut être traitée de façon centralisée, avec par exemple un évaluation multi-critères, ou comme nous le proposons, par la combinaison de plusieurs évaluations propres aux agents du système.

La suite de ce paragraphe présente les problématiques que nous avons identifiées lors de la conception de notre système de contrôle. Certains points apportent des pistes de solutions que nous approfondirons dans de futurs travaux.

8.2.1 Correction d'une erreur ou d'une défaillance

La correction d'une erreur ou d'une défaillance intervient suite à la réception d'un message d'alerte envoyé par la tâche *Robustesse*. Ce message est composé de deux parties : une pour l'élément identifié comme l'origine de l'erreur ou de la défaillance, et une pour le degré de confiance dans cet événement. Un degré de confiance maximal implique que la cause de l'alerte est avérée et qu'il est impossible de l'éviter. Si, au contraire, l'erreur ou la défaillance n'est pas avérée, nous définissons deux étapes lors de la prise de décision suite à une alerte :

1. l'évaluation du besoin de reconfiguration ;
2. l'évaluation de l'action à appliquer.

Ainsi, nous proposons que le système de contrôle effectue un évitement d'erreur ou de défaillance lorsque l'événement n'est pas avéré, ou un recouvrement après l'occurrence de l'événement.

Évitement

L'évitement d'une erreur ou d'une défaillance n'est pas toujours possible et le lancement d'une reconfiguration, alors que l'événement n'est pas avéré, est toujours basé sur une évaluation de l'état futur du système.

Cette évaluation est généralement de nature imprécise ou incertaine. Nous prendrons pour exemple les deux pronostics de défaillance suivants :

« Le système va certainement tomber dans un état de défaillance
dans les 24 prochaines heures. »

et
« La probabilité que le système soit défaillant
à la 42^{ieme} seconde de la prochaine heure est faible. »

Intuitivement le pronostic d'occurrence d'une erreur est imprécis ou incertain, et la prise de décision doit alors prendre en compte ces deux composantes dans l'évaluation du besoin de reconfiguration.

Dans l'état actuel de nos travaux, nous supposons disponible(s) la (les) fonction(s) multi-critères d'évaluation du besoin de reconfiguration. La nuance que nous souhaitons mettre en avant dans nos prochains travaux concerne l'évaluation d'une situation selon plusieurs fonctions multi-critères. En effet, notre approche base ce mécanisme d'évaluation sur une négociation menée au sein de la communauté d'agents qui composent notre système. Comme nous le détaillons dans le Chapitre 9, notre système de contrôle est basé sur deux communautés d'agents, dont une est responsable, entre autres, de l'exécution du mécanisme de prise de décision.

L'idée qui guide notre approche réside sur une des caractéristiques majeures de notre système : il n'y a pas d'élément central ou d'information globale sur le système. Dès lors que nous supposons qu'aucune vision globale du système n'est disponible, il est raisonnable de penser que plusieurs visions partielles du système peuvent être établies. De plus, si la fonction d'évaluation d'un besoin de reconfiguration peut être appliquée sur une vue « topologique » limitée du système, elle doit également être applicable sur un historique incomplet de connaissances formulées sur le système. En effet, la place d'un agent au sein du système de contrôle peut évoluer dans le temps et le passé de chaque agent devient alors unique. Cette individualisation des agents de la communauté amène le second axe de notre idée, où la prise de décision est basée sur une fonction d'évaluation unique, applicable sur une donnée limitée et un historique de connaissances incomplet.

L'intérêt d'une étape de négociation entre les agents devient alors pertinent et nous analyserons, dans de prochains travaux, la faisabilité d'une telle approche dans le cadre du contrôle des systèmes de fusion d'informations par des systèmes multi-agents.

La seconde étape de prise de décision, identifiée comme l'évaluation de l'action à appliquer suite à une alerte, est commune à l'évitement et au recouvrement. Nous détaillons cette étape dans le paragraphe suivant.

Recouvrement

Le recouvrement, à la différence de l'évitement, n'a lieu que suite à l'occurrence d'une erreur ou d'une défaillance ; le message d'alerte transmis est donc précis et certain. Pour autant, certains raccourcis sont à éviter car des problèmes peuvent survenir même lors de la réception d'un tel message d'alerte. En effet, le mécanisme de prise de décision doit être en mesure d'identifier les fausses alertes. Dans nos travaux, il sera donc important d'identifier quelles sont les vérifications à effectuer pour prévenir un recouvrement basé sur une fausse alerte.

L'étape d'évaluation du besoin de reconfiguration, nécessaire dans le cas de l'évitement, est ici inutile car la défaillance ou l'erreur qui va mener à une défaillance, est avérée. Ainsi, dès que le choix d'une reconfiguration est pris par le système de contrôle, celui-ci doit être en mesure de diagnostiquer quels sont les éléments de la configuration à adapter. La plus simple des modifications consiste à remplacer l'élément erroné ou défaillant par un autre élément. Trois actions sont alors possibles :

- le redéploiement à l'identique de l'élément en question, ce qui consiste en un redémarrage d'une partie de la configuration ;
- le déploiement d'une autre entité d'exécution sans modifier l'affectation des fonctions de fusion ;

– la modification de l'affectation des fonctions de fusion ou de celle des liens entre les fonctions.

Ces trois actions concernent l'élément identifié dans le message d'alerte mais d'autres éléments de la configuration du système peuvent également être modifiés. En effet, l'action sélectionnée pour corriger l'état de défaillance du système peut elle-même invalider une autre partie de la configuration.

Ce que nous proposons dans un premier temps, est d'appliquer ce mécanisme de recouvrement itérativement sur le système. Ainsi, si la correction d'une erreur induit une nouvelle erreur dans le système, celle-ci devrait être détectée puis à son tour corrigée.

D'autres solutions sont également envisagées, comme la création d'un modèle de la configuration corrigée du système. Ce modèle serait alors analysé par la tâche de contrôle *Analyse* et une éventuelle erreur introduite par le recouvrement serait détectée puis corrigée.

Ces deux approches partagent le même but et impliquent un intervalle de temps de correction qui peut être significativement pénalisant. Ainsi, le temps avant le déploiement d'une configuration « saine » dépend du nombre de nouvelles sources d'erreurs introduites puis de leur détection, ou de la durée de l'analyse du modèle du système.

Si notre étude de ces approches souligne que les temps de correction ne sont pas significativement différents, alors le choix d'une des deux approches sera guidé par le « risque », ou la pénalité, induite par le déploiement de configurations erronées pendant la phase de correction.

8.2.2 Amélioration de la configuration

L'amélioration de la configuration est la seconde cause de reconfiguration du système. Elle a lieu lors de la découverte conjointe d'une meilleure configuration par le mécanisme de parcours de l'espace des configurations et par le mécanisme d'analyse du modèle du système. Nous proposons que cette amélioration soit formulée sur les indices de performance calculés à partir du modèle d'analyse.

Meilleure configuration

Le Chapitre 7 détaille l'analyse menée sur un modèle de la configuration du système. Cette analyse nous permet de calculer des indices de performances, que nous proposons d'agréger [BHMP09] pour élaborer un gain que le système de contrôle doit optimiser.

À l'instar de l'évaluation de la nécessité d'une reconfiguration suite à la réception d'une alerte, nous proposons pour les mêmes raisons, que l'évaluation du gain d'une reconfiguration soit basée sur un mécanisme de négociation entre les agents du système. Ainsi, ces agents disposeraient de fonctions d'évaluation sensiblement différentes, au niveau de leur formulation ou au niveau de leur donnée d'entrée.

Configuration courante mal évaluée

Suivant la même idée que le paragraphe précédent, si la configuration courante n'atteint pas les performances initialement planifiées, alors le système de contrôle évalue le potentiel gain d'une reconfiguration.

Par ailleurs, le déploiement d'une configuration mal évaluée signifie que le modèle du système a évolué. Comme la structure de celui-ci est dérivée de la configuration actuelle du système, il est clair que les seules évolutions possibles concernent les attributs de ce modèle. Ainsi, les attributs du modèle d'analyse sont corrigés pour correspondre aux nouvelles caractéristiques du système. De là, de nouvelles configurations qui étaient écartées dans un premier temps, peuvent se

révéler plus performantes que celle actuellement déployée. Le gain d'une reconfiguration sera donc également évalué entre des configurations candidates et la configuration courante du système.

8.2.3 Gestion de la reconfiguration

Le mécanisme de prise de décision s'inscrit dans le système de contrôle global et dépend ainsi d'autres tâches de contrôle déjà présentées. Les trois interactions possibles sont basées sur des échanges de messages entre la tâche *Reconfiguration*, qui héberge le mécanisme de prise de décision, et les tâches *Analyse*, *Robustesse* et *Configuration*.

Ces trois autres tâches de contrôle sont à l'interface entre le sommet du système de contrôle et les actionneurs, ou sources de mesures, installés dans le système de fusion. À ce titre, nous devons traduire les actions sélectionnées par le mécanisme de prise de décision, dans des commandes compatibles avec la tâche *Configuration*, seul intermédiaire possible pour la reconfiguration du système.

Nouveau diagnostic du système

Le diagnostic du système, fourni par la tâche *Robustesse*, établit d'une part un pronostic des éventuelles erreurs ou défaillances du système, mais donne également une mesure des performances de la configuration courante. Ainsi, à partir de ces données, un gain de reconfiguration est évalué et une action à appliquer sur le système est sélectionnée. Celle-ci correspond à une modification de la configuration actuelle, que nous détaillons ci-dessous.

Nouveaux indices de performance

De nouveaux indices de performance peuvent être envoyés par la tâche *Analyse* à la tâche *Reconfiguration*. Ces indices sont toujours associés au modèle utilisé et il est ainsi possible d'identifier la configuration à l'origine de l'analyse. De là, ces indices sont confrontés aux mesures de performance réalisées sur le système réel, et une reconfiguration du système est initiée si la configuration évaluée est plus performante que celle actuelle.

Modification de la configuration actuelle

Cette dernière interaction définit l'action à appliquer sur la configuration courante. Cette action peut être de trois natures comme nous le présentons dans le Paragraphe 8.2.1, et elle doit être adaptée aux données acceptées par la tâche *Configuration*.

Le redéploiement d'une partie de la configuration s'effectue par la réinitialisation du parcours de l'espace de recherche à partir de la configuration courante. En effet, dès que cette étape est franchie, la tâche *Configuration* ordonnera le déploiement de cette configuration et les éléments erronés seront redémarrés.

L'échange d'entité d'exécution se traduit par la suppression de la relation de compatibilité entre l'actuelle implémentation déployée et le cadre d'exécution auquel elle est affectée. Cette suppression correspond à la suppression d'une partie de l'espace de recherche de la tâche *Configuration*. Une nouvelle entité d'exécution est alors sélectionnée par cette tâche de contrôle.

Finalement, la modification de l'affectation des fonctions de fusion, ou de celle des liens entre les fonctions, correspond à la suppression d'arcs dans le graphe biparties qui décrit l'espace des configurations possibles. La découverte de nouvelles ressources est au contraire traduite par l'ajout de tels arcs.

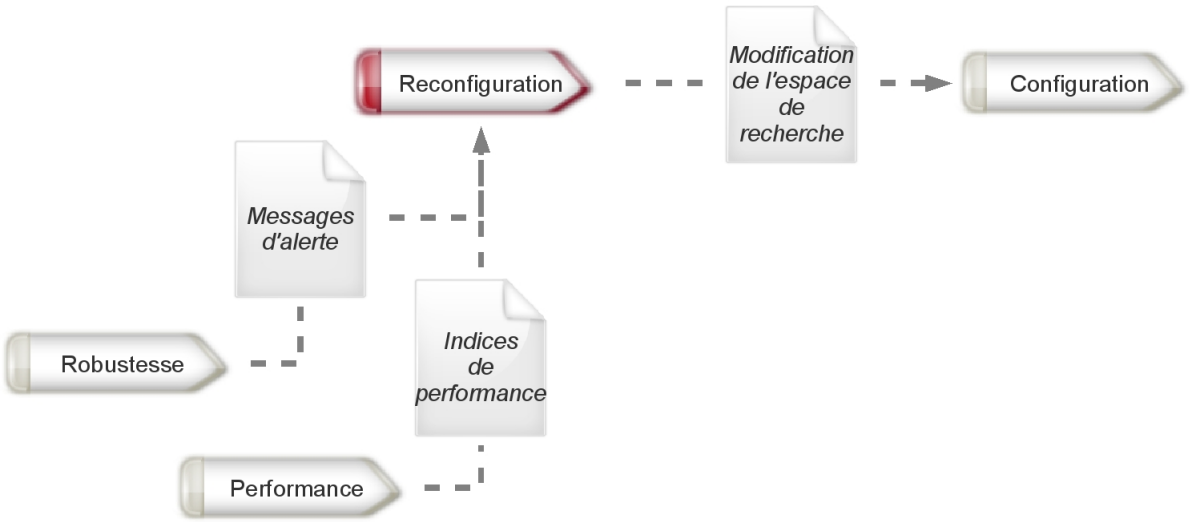


FIG. 8.5 – Tâche de contrôle liée à la reconfiguration.

8.3 Synthèse

Lors de notre conception d'un système de contrôle pour les SRFI, nous avons identifié plusieurs problématiques, notamment la nécessaire prise de décision relative au déclenchement d'une reconfiguration. Les pistes de solutions que nous proposons et que nous avons en partie abordées, ne présentent pas toutes les deux le même intérêt.

Certes l'application d'un mécanisme de fouille de données au pronostic de défaillances est un apport original dans la communauté des systèmes répartis de fusion. De plus, de part sa définition, cet algorithme permet la surveillance du système selon des règles de prédictions différentes. En effet, l'historique de mesures sur lequel est basé chaque pronostic, est différent pour chaque agent du système. Ainsi, même si les valeurs des sondes surveillées et l'algorithme déroulé sont les mêmes, les connaissances de chaque agent seront différentes. De cette multiplicité des diagnostics découle une problématique plus prometteuse en terme d'avancées.

Ainsi, le choix d'une reconfiguration du système, induite par une possible défaillance, ou pour une amélioration des performances, nécessite une évaluation de la situation et une prise de décision. Ces deux étapes, qui définissent d'une part un problème de fusion, et d'autre part un problème de décision, pourraient être mises en œuvre grâce à un protocole ou un mécanisme basé sur un système d'agents. Selon nous, plusieurs choix sont alors disponibles : faut-il que les agents partagent leurs points de vue avant de prendre leur décision, ou faut-il au contraire que chaque agent garde ses connaissances et infléchisse la décision de la communauté ? Les outils manipulés à ce niveau d'abstraction sont totalement différents de ceux que nous avons exploités jusqu'à présent. Sans doute faudra-t-il combler ce vide, par des compléments de recherche ou la proposition de nouveaux systèmes, avant d'atteindre ce niveau d'intelligence de la part des agents.

Chapitre 9

Répartition du contrôle

Ce chapitre du mémoire présente notre modèle de répartition du système de contrôle que nous avons défini dans le Chapitre 5. Notre approche se base sur un Système Multi-Agents (SMA) composé d'agents réactifs qui implémentent un mécanisme de différentiation bio-inspiré. Ce mécanisme, ainsi que d'autre plus conventionnels, sont présentés dans la suite de ce chapitre. Le Paragraphe 9.1 présente ainsi, un panorama des méthodes de répartition les plus courantes, comme par exemple le parallélisme de données ou les approches hiérarchiques. Nous faisons en suite un rappel des définitions du domaine des SMA dans le Paragraphe 9.2 puis nous présentons le cœur de notre approche dans les Paragraphes 9.4 à 9.9.

9.1 Les types de répartition

En prémisse de la présentation de notre mécanisme, nous proposons un rappel des définitions des termes souvent rencontrés, et acceptées par la communauté des systèmes répartis.

Parallélisme Le but du parallélisme est de décomposer un problème donné pour le résoudre avec plusieurs ordinateurs afin de diminuer le temps de traitement global. Plusieurs types de parallélisme sont possibles :

- le parallélisme de contrôle ;
- le parallélisme de flux ;
- et le parallélisme de données.

Le parallélisme de contrôle, illustré sur la Figure 9.1, permet la décomposition d'une application en différentes tâches qui seront confiées à différents processeurs. Deux cas sont envisagés, suivant que les tâches à exécuter sont dépendantes ou non. Si l'on pose T_i le temps d'exécution de la tâche i, le temps d'exécution total est $T_{total} = \sum_i T_i$ dans le cas d'une exécution séquentielle et $T_{total} = max\{T_i\}$ dans le cas d'un parallélisme de contrôle où toutes les tâches sont indépendantes. L'exemple de la Figure 9.1 illustre un cas où deux tâches, t_2 et t_3, dépendent de la tâche t_1 ; le temps d'exécution est alors $T_{total} = T_1 + max\{T_2, T_3 + T_4\}$.

Le parallélisme de données, illustré sur la Figure 9.2, propose l'exécution des mêmes tâches sur des données divisibles. Dans ce type de parallélisme, les données sont réparties sur les différents processeurs ce qui peut dans certains cas être difficile, principalement à cause des dépendances entre les parties des données. Le parallélisme de données permet un temps d'exécution total $T_{total} = max\{T_i\}$ si les processeurs ne sont pas équivalents.

Le parallélisme de flux, illustré sur la Figure 9.2, et également appelé *Pipeline*. Ce type de parallélisme permet la décomposition d'un tâche en sous-tâches successives. Si l'on pose T le temps d'exécution de chaque sous-tâches, n le nombre de données à traiter et K le nombre

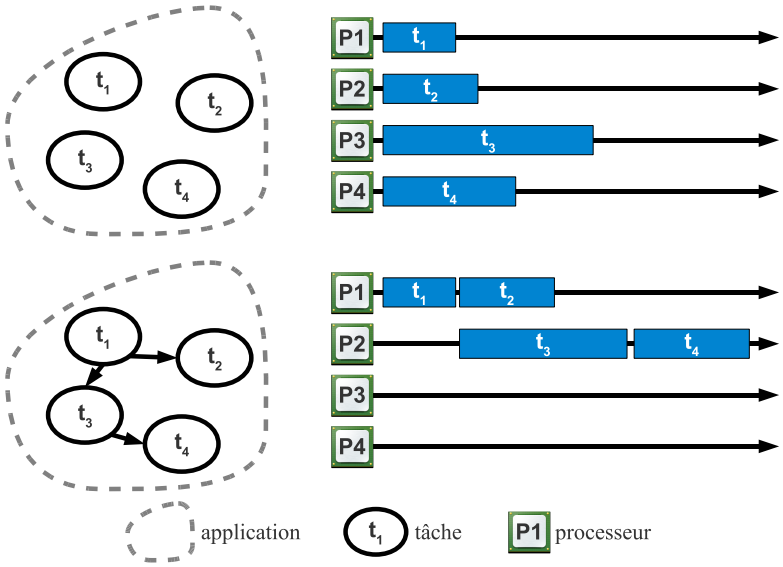


FIG. 9.1 – Les différents types de répartition : le parallélisme de contrôle sans dépendance et avec dépendance.

de sous-tâches, alors le temps d'exécution total en séquentiel est de $T_{total} = K \times n \times T$ et $T_{total} = ((K-1)+n) \times T$ avec un parallélisme de flux.

Hiérarchique L'approche hiérarchique est un mélange du parallélisme de flux et de données. Dans ce type de répartition, l'application est divisée en niveaux de traitement, d'où le parallélisme de flux, où chaque niveau est divisé en sous-tâches équivalentes qui traitent les données issues des niveaux précédents.

Réplication La dernière approche de répartition possible est la réplication dont le but est d'améliorer la disponibilité d'une application. Ainsi, comme l'illustre la Figure 9.3, la même application est installée sur plusieurs ordinateurs et une donnée à traiter est affectée à une des instances disponibles.

9.2 Système multi-agents

9.2.1 Définition

La définition, la plus souvent rencontrée, d'un système multi-agents est la suivante : *« Un système multi-agents (SMA) est un système composé d'un ensemble d'agents, situés dans un certain environnement et interagissant selon certaines relations. Un agent est une entité caractérisée par le fait qu'elle est autonome »* .

Cette définition très générale permet de cerner la nature d'un agent, mais elle ne répond pas à la question « Pourquoi utiliser un SMA ? ». Des éléments de réponse foisonnent dans les publications scientifiques et, loin d'adopter une vision extrémiste de l'utilité des SMA, notre point de vue concorde avec celui présenté dans [Fer97]. Ainsi, un SMA répond à une catégorie de problèmes caractérisés par les points suivants :

– les problèmes sont physiquement distribués : la résolution du problème fait intervenir un large espace où les ressources sont localisées à différents emplacements, comme par exemple

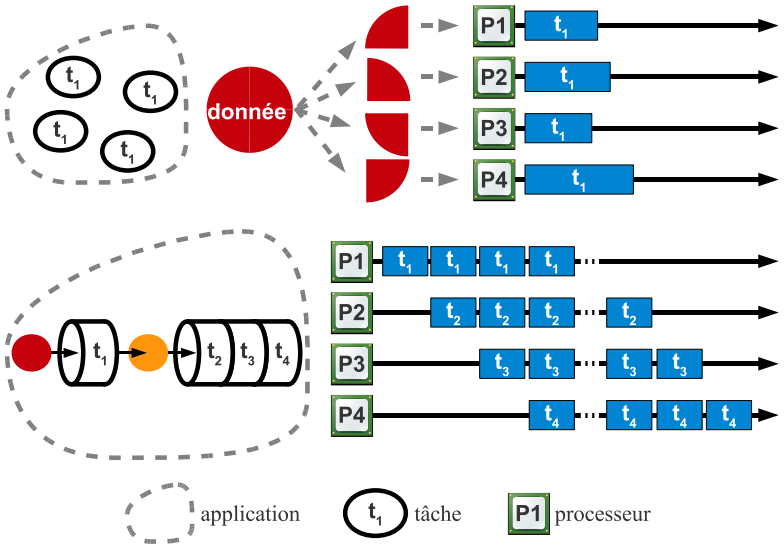


FIG. 9.2 – Les différents types de répartition : parallélisme de donnée et parallélisme de flux.

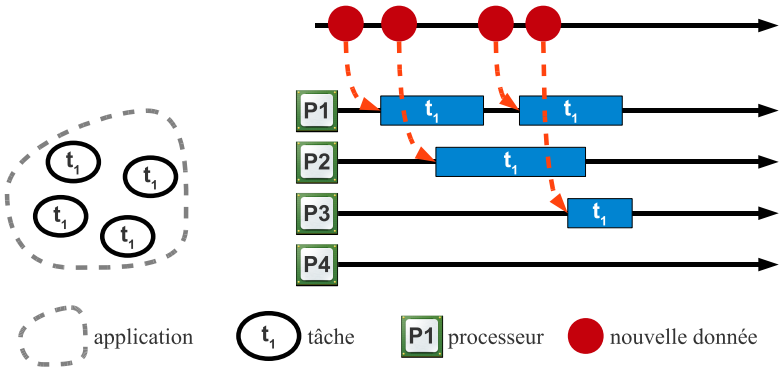


FIG. 9.3 – Les différents types de répartition : la réplication.

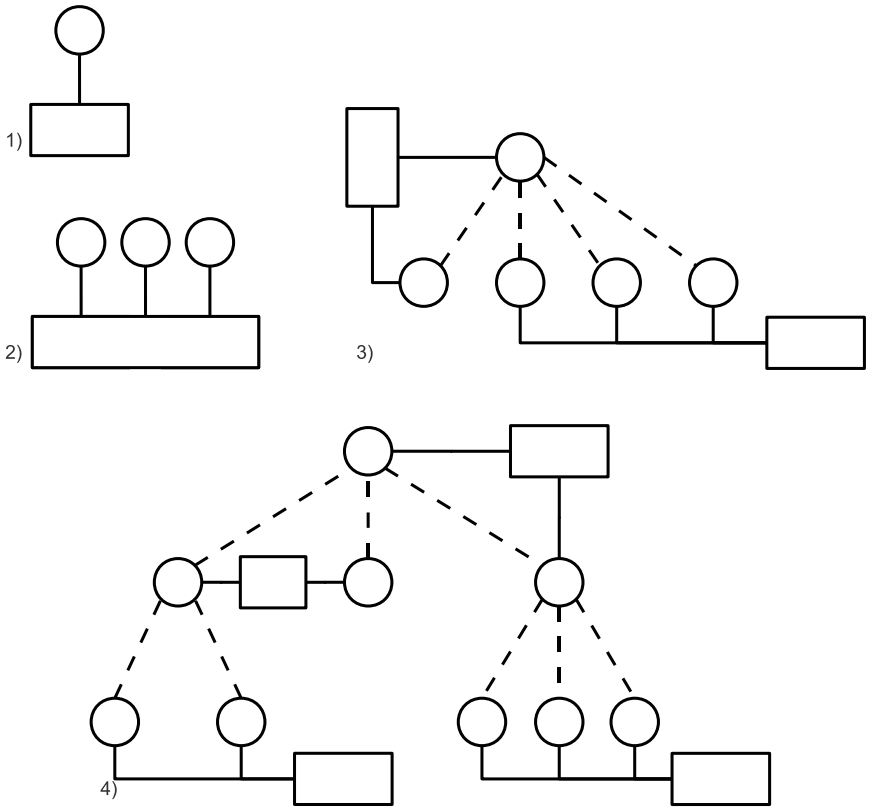


FIG. 9.4 – Structures des SMA [BD95] : (1) groupe simple, (2) équipe, (3) hiérarchie simple et (4) hiérarchie multi-niveaux.

une chaîne de production ;
- les problèmes sont fonctionnellement distribués et hétérogènes : il n'existe pas un unique expert capable de résoudre tous les aspects du problème. En outre, au-delà des compétences requises, l'utilisation conjointe de plusieurs outils peut être nécessaire ;
- les réseaux impliquent une vision distribuée : les systèmes actuels ne peuvent plus être basés sur une vision globale et l'interopérabilité dans un système ouvert est incontournable. Les SMA sont une solution de construction de systèmes hétérogènes et souples sans structure *a priori* ;
- la complexité du problème impose une vision locale : certains problèmes sont trop vastes pour être résolus globalement et une « pensée locale » permet de simplifier l'obtention des résultats en évitant les difficultés liées à l'inefficacité d'algorithmes globaux, tout en offrant des résultats comparables qui émergent des interactions locales ;
- les systèmes doivent pouvoir s'adapter à des modifications de structure et d'environnement : la complexité des systèmes informatiques impose la conception de logiciels adaptables aux modifications du contexte de travail et à l'évolution des besoins ;
- le génie logiciel va dans le sens d'une conception en termes d'unités autonomes en interactions : les systèmes informatiques intègrent à présent la distribution des traitements et l'hétérogénéité des réalisations.

Plus formellement, nous adoptons la Définition 9.2.1, proposée dans [Fer97].

Définition 9.2.1 (Agent). *On appelle agent une entité physique ou virtuelle*
- *qui est capable d'agir dans un environnement ;*
- *qui peut communiquer directement avec d'autres agents ;*
- *qui est mue par un ensemble d'objectifs individuels ;*
- *qui possède des ressources propres ;*
- *qui est capable de percevoir son environnement ;*
- *qui ne dispose que d'une représentation partielle de son environnement ;*
- *qui possède des compétences et offre des services ;*
- *qui peut éventuellement se reproduire ;*
- *dont le comportement tend à satisfaire ses objectifs ; en tenant compte des ressources et des compétences dont elle dispose ; et en fonction de sa perception, de ses représentations et des communications qu'elle reçoit.*

9.2.2 Utilité d'un système multi-agents

Dans le Chapitre 4, nous dressons les différentes approches qui peuvent utiliser conjointement les SMA et les SRFI :
- des agents peuvent mettre en œuvre un processus de fusion et coordonner plusieurs points de vue pour en faire émerger un résultat ;
- un agent peut utiliser un système de fusion d'informations pour élaborer des connaissances qui lui sont propres, lui permettant par la suite de prendre des décisions ;
- enfin des agents peuvent être utilisés pour contrôler l'exécution d'un processus de fusion d'informations.

Nous souhaitons aborder ce dernier point dans la suite de ce paragraphe, car l'utilisation d'un SMA est dans notre cas justifiée par le caractère réparti, dynamique, et ouvert de notre système de contrôle. Lors de notre recherche d'une solution de répartition, deux points ont attiré notre attention.

Les SMA répondent à des structures ou organisations connues et déjà éprouvées. [BD95] dresse un panorama de ces organisations dont nous présentons une sélection dans la Figure 9.4 : un cercle représente un agent, un rectangle un dépôt d'informations, une ligne pleine définit une

commande et une ligne discontinue une information. Cette sélection reprend deux cas simples et deux cas applicables au contrôle des SFI. Nous jugeons les structures plus complexes, comme celles basées sur la programmation orientée-marché, inappropriées pour la mise en œuvre de notre système de contrôle Les structures (3) et (4) de la Figure 9.4 sont les deux alternatives adaptables à notre approche pour deux raisons. Premièrement, de part la complexité des tâches de contrôle que nous mettons en œuvre, la division du problème en sous-problèmes au sein d'un niveau de l'arborescence est pertinente. La nature de certaines tâches de contrôle nous permet d'appliquer un contrôle sur un sous-ensemble du système ce qui réduit le besoin d'une vue globale du SFI.

Au même titre que les organisations de SMA peuvent être exploitées pour la structure de la répartition du système, d'autres solutions peuvent être exploitées. C'est le cas des protocoles d'interaction entre les agents d'un SMA, utilisés lors de la mise en œuvre de nos mécanismes de contrôle. Ainsi, le protocole *contract net*, proposé en 1980 dans [Smi80] puis redéfini dans [Fou02], est toujours d'actualité dans la communauté agent. La tâche de contrôle *Configuration* tire la définition de l'espace de recherche de configuration du résultat de ce protocole. En effet, les agents du système présentent les ressources dont ils disposent sous la forme d'un contrat où chaque ressource offerte par un agent est notée dans un message envoyée au sein de la tâche *Configuration*. La quantité des ressources communiquée pendant cette phase, est alors basée sur les mesures issues des sondes installées dans le système.

9.3 Contrôle hiérarchique

Nous avons sélectionné une approche hiérarchique pour répartir les tâches de contrôle *Configuration* et *Reconfiguration*. Le hiérarchie est ici définie au sein d'une tâche de contrôle, comme l'illustre la Figure 9.5 et non entre les tâches du système, comme le présente la Figure 9.7. Ce choix est motivé par la nature de ces tâches qui peuvent être appliquées à des échelles différentes. En effet, la tâche *Configuration* parcourt un espace des configurations possibles et sélectionne une configuration candidate pour les autres tâches de contrôle. De là, l'espace de recherche peut être découpé en sous-espaces construits à partir des données issues d'un groupe d'agents. Ce découpage permet donc un parallélisme de données et la recherche effectuée à un niveau fixé peut être réutilisée à un niveau supérieur, d'où le parallélisme de flux. L'utilisation d'une répartition hiérarchique se justifie de la même façon pour la tâche de contrôle *Reconfiguration*.

Hiérarchie multi-niveaux L'organisation des agents que nous proposons au sein d'une tâche de contrôle correspond à la définition d'une hiérarchie multi-niveaux [BD95]. Ainsi, la relation de contrôle, illustrée par la Figure 9.5, forme un arbre où l'information est échangée seulement entre des niveaux adjacents. Les agents se trouvant à un certain niveau adoptent le rôle de superviseur vis-à-vis des agents du niveau inférieur. Le contrôle est ainsi réparti sur l'ensemble des niveaux et chaque agent ne traite qu'une partie du problème : soit par la division de la donnée à traiter, soit par le pré-traitement effectué aux niveaux inférieurs.

9.4 Répartition par tâche

Notre système de contrôle est composé de plusieurs tâches de contrôle réparties selon une hiérarchie multi-niveaux. Les tâches de contrôle parallélisées sont caractérisées par leur place dans le système de contrôle. En effet, si l'on reprend les Figures 5.1 et 5.2 et que l'on positionne les tâches de contrôle comme l'illustre la Figure 9.7, nous pouvons reformuler l'organisation du système de contrôle selon trois couches : la couche de contrôle qui permet l'interaction avec le système de fusion, celle d'agrégation et celle de décision.

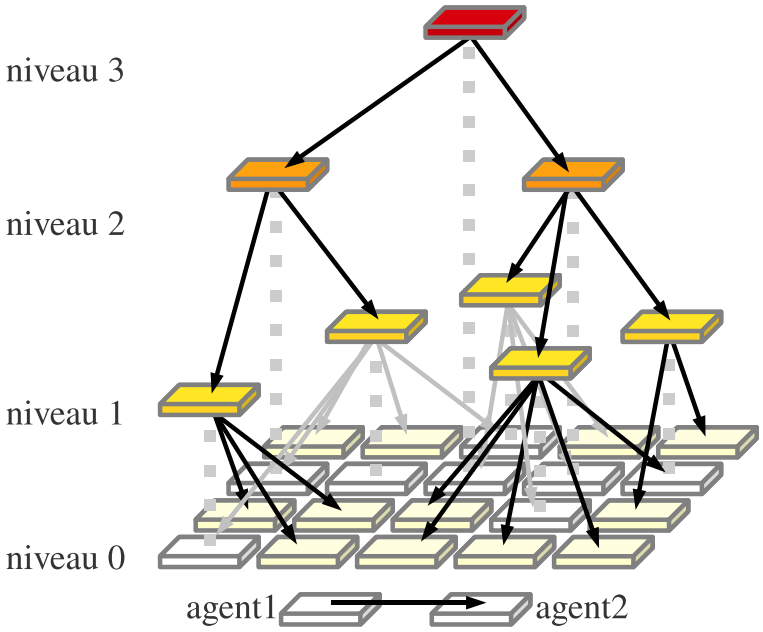


FIG. 9.5 – Tâche de contrôle hiérarchique : agent1→agent2 représente la relation « agent1 exploite les données de l'agent2 ».

Les deux premiers niveaux sont parallélisés alors que le dernier suit une organisation hiérarchique. Les motivations de ce choix ont été présentées dans le Paragraphe 9.3.

Exemple Pour illustrer le fonctionnement de notre structure de contrôle, considérons la tâche en charge de la recherche d'une configuration. Les agents d'un niveau donné vont gérer la construction de sous-espaces de recherche, définis par les groupes d'agents qu'ils contrôlent. La charge de calcul est alors divisée selon un parallélisme de donnée, où la donnée à traiter est répartie selon les agents du système.

L'autre avantage d'une répartition hiérarchique tient dans le traitement déjà effectué dans les niveaux inférieurs. Si l'on reprend le cas d'une recherche de configuration, le parcours de l'espace des solutions pour un agent et un niveau donné, écartera les sous-espaces déjà parcourus. Un agent doit alors rechercher une configuration en considérant les données qu'il reçoit des niveaux inférieurs et restreindre sa recherche à un niveau global.

9.4.1 Parallélisme

La première couche du système de contrôle possède deux interfaces. En effet, comme l'illustre la Figure 9.7, la partie basse du système de contrôle est connectée aux sources d'informations et aux actionneurs du SFI. Ainsi, c'est dans cette couche de contrôle qu'a lieu le déploiement effectif d'une configuration.

D'autre part, le haut de cette couche est à l'interface entre le système de fusion et le système de contrôle. À ce titre, ce niveau de contrôle propose également des sources d'informations, appelées sondes pour éviter les confusions, et des actionneurs qui permettent d'agir sur la configuration du déploiement.

Cette première couche de contrôle regroupe au sein de la tâche *Implémentation* le lien entre le système de fusion et notre mécanisme de contrôle. Une extension de notre système pour un autre

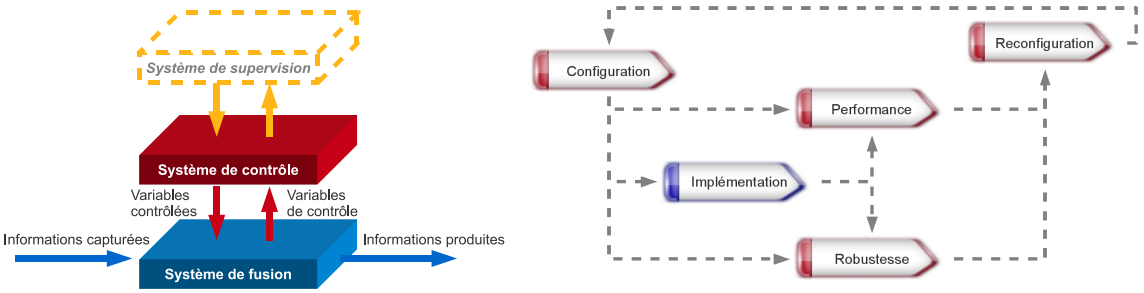


FIG. 9.6 – Organisation du système de contrôle (Figure 5.1) et détail de la boucle de contrôle avec les quatre tâches du système de contrôle (Figure 5.2) .

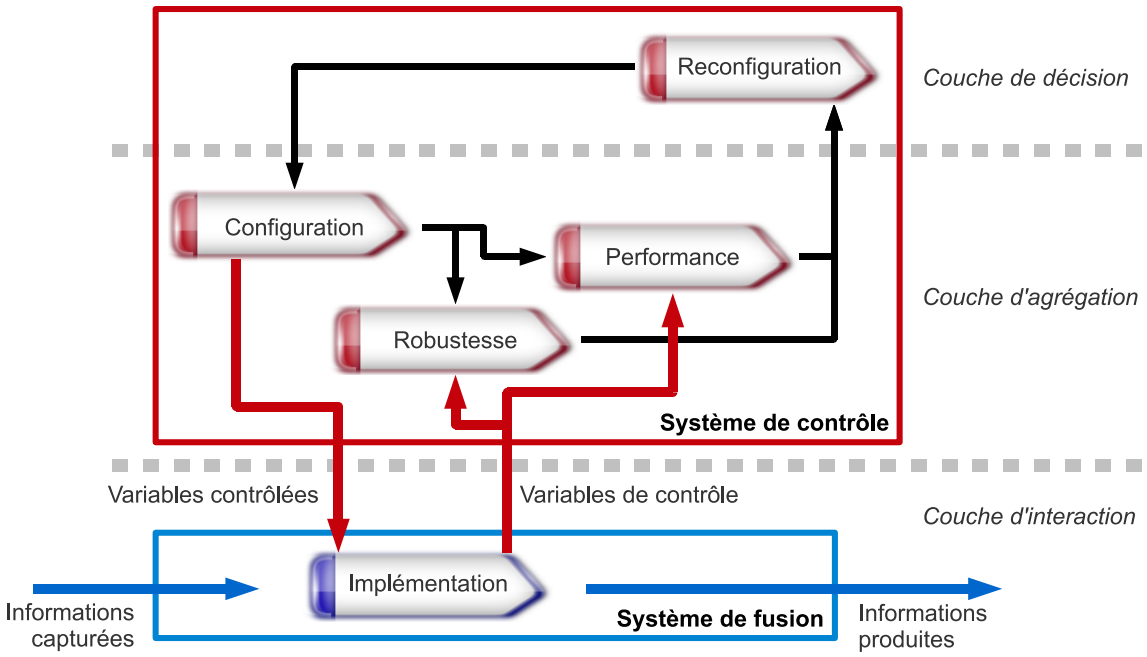


FIG. 9.7 – Positionnement des tâches de contrôle au sein des systèmes de contrôle et d'exécution.

type de système de fusion, comme par exemple Global Sensor Network [Sal07], s'accompagne du développement d'une autre implémentation de cette tâche de contrôle.

Le parallélisme de données s'impose ici de part la répartition physique des agents qui hébergent le système de fusion. Ainsi, comme l'illustre la Figure 9.8, une instance de la tâche *Implémentation* est activée sur chaque agent du système et ne traite que les données relatives à cet agent. Cette tâche ne nécessite qu'une vue locale du système et il est donc inutile de l'organiser sur plusieurs niveaux.

La seconde couche de contrôle (Figure 9.7) est composée des trois tâches de contrôle dont deux sont réparties selon un parallélisme de données. Le mécanisme de répartition de la tâche *Configuration* est présenté dans le paragraphe suivant et nous ciblons ici celui des tâches *Robustesse* et *Performance*. Celles-ci sont parallélisées dans le système sur deux niveaux. Le premier niveau est activé par tous les agents et les tâches de contrôle récupèrent les mesures fournies par les sondes de la tâche *Implémentation*. Le second niveau traite effectivement les mesures pour des groupes d'agents et fournit des informations issues de l'agrégation des mesures produites par

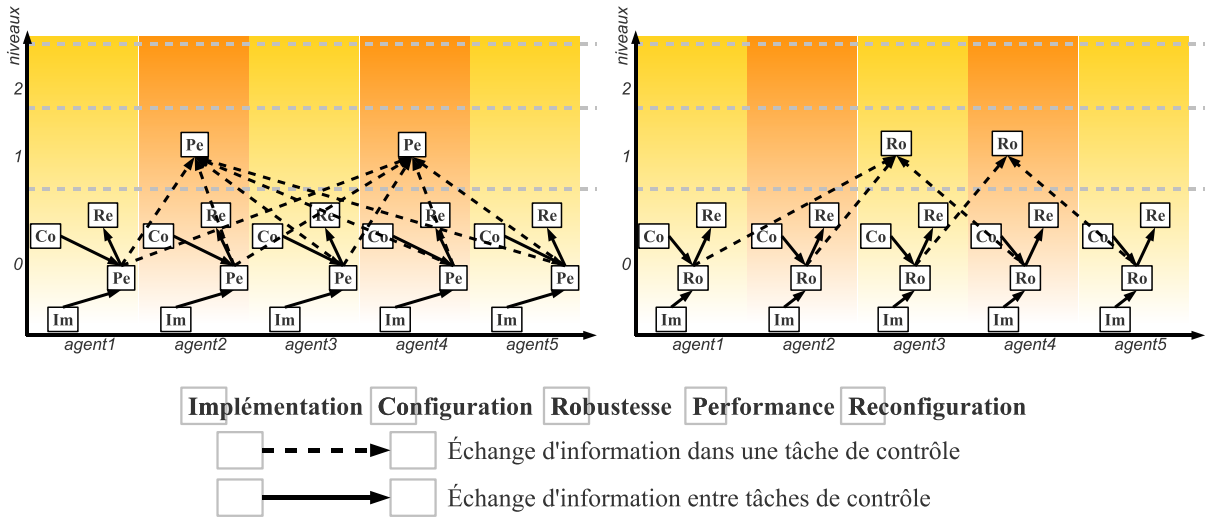


FIG. 9.8 – Les tâches de contrôle et leurs niveaux dans la hiérarchie : cas des tâches *Implémentation*, *Performance* et *Robustesse*.

la couche d'interaction.

Quelques contraintes sont définies sur l'organisation de la répartition des tâches de cette couche de contrôle. Dans notre approche, la tâche *Robustesse* est chargée de la surveillance du système. À ce titre un agent ne peut être responsable de sa surveillance ce qui implique l'activation d'au moins deux instances de cette tâche de contrôle au second niveau.

L'activation d'une instance de la tâche *Performance* au second niveau sera quant à elle uniquement établie en fonction de la puissance de l'agent qui l'héberge.

Concernant les échanges d'informations au sein des tâches *Robustesse* et *Performance*, une hiérarchie sur tout le système est parfois justifiée. En effet, la tâche *Robustesse* ne nécessite qu'une vision partielle de la communauté d'agents : précisément un groupe par agent en charge de la surveillance. La tâche *Performance* nécessite au contraire une vue globale du système car le but de ces communications est la mise à jour des attributs du modèle d'analyse traduit à partir de la configuration du SFI.

9.4.2 Hiérarchie multi-niveaux

La troisième tâche de contrôle de la seconde couche et celle de la dernière couche de contrôle, illustrée par la Figure 9.7, sont réparties selon une hiérarchie multi-niveaux.

Les tâches *Configuration* et *Reconfiguration* sont elles aussi activées sur tous les agents du système, comme l'illustre la Figure 9.9. De part leur nature, ces tâches de contrôle peuvent être appliquées à plusieurs échelles. Ainsi, une répartition en largeur permet de traiter une quantité de données plus importante - ici directement liée à la taille de la population d'agents - alors qu'une répartition multi-niveaux permet un pré-traitement d'un niveau sur les niveaux supérieurs. Un tel pré-traitement est par exemple utile lors de la recherche d'une configuration dans l'espace des solutions où les régions déjà explorées ne sont plus traitées.

La tâche *Reconfiguration*, en charge de l'étape de prise de décision, est répartie selon le même principe. En effet, il est inutile de faire remonter une requête au sommet de l'arborescence si elle peut être traitée à un niveau intermédiaire.

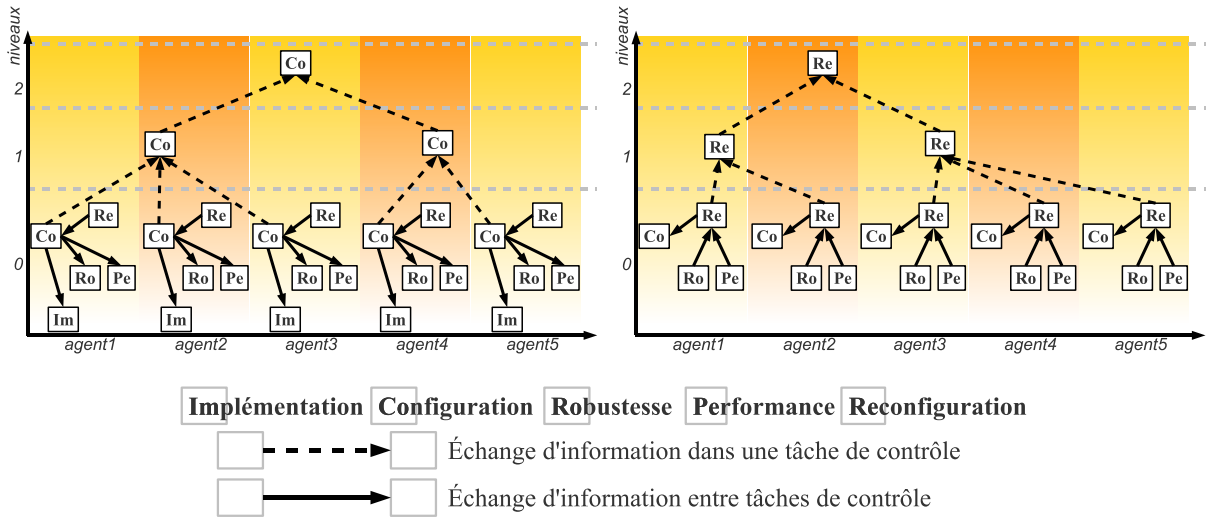


FIG. 9.9 – Les tâches de contrôle et leurs niveaux dans la hiérarchie : cas des tâches *Implémentation*, *Configuration* et *Reconfiguration*.

9.5 Deux types d'agents

Motivations Parmi les architectures de système que nous visons (listées dans le Chapitre 2.2.2) nous distinguons deux types de cadres d'exécution. Les premiers ont une puissance de calcul limitée et une connexion réseau intermittente ou limitée. Ce groupe correspond principalement aux PDA, aux *smart-phone* et aux PC portables. Les autres architectures, comme par exemple les PC de bureau ou les *box* Internet, sont caractérisées par une importante puissance de calcul et une connexion réseau permanente.

En conséquence de l'identification de ces deux types de cadres d'exécution, nous proposons la définition d'un type d'agent pour chaque famille d'architectures. Les agents les plus puissants sont dits « immobiles », et ont la charge du contrôle du système. Leur connexion permanente au réseau facilite les échanges de messages de contrôle, notamment avec le second type d'agents. Ces derniers agents, sont dits « mobiles » proposent leurs ressources aux agents immobiles lors du déploiement d'une configuration. Même si les ressources des agents sont plus modestes que celles des agents immobiles, leur proximité vis-à-vis de l'utilisateur les rends indispensables : par exemple un agent mobile hébergé sur un PDA.

9.5.1 Agent mobile

Définition 9.5.1 (Agent mobile). *Un agent mobile*[1] *est associé aux architectures qui ont une puissance de calcul limitée et une connexion réseau limitée. Un agent mobile est un agent réactif et interagit avec une communauté d'agents immobiles lors de la réception d'une requête de contrôle.*

Un agent mobile possède trois objectifs :
- la découverte d'une communauté d'agents immobiles ;
- le traitement d'informations ;
- l'application des requêtes de contrôle.

Le premier objectif sert à l'intégration d'un agent mobile dans le système. Dès cette étape franchie, l'agent mobile remplace cet objectif par les deux suivants pour activer une partie du

[1] Les adjectifs « mobile » et « immobile » utilisés dans la suite de ce chapitre sont inspirés du roman [Ham06].

système de fusion et une partie du système de contrôle. En effet, chaque agent du système active le niveau le plus bas de toutes les tâches de contrôle ce qui permet au système de contrôle de toujours être en mesure de déployer une partie de la configuration sur un agent et de détecter une modification des ressources.

En conséquence des objectifs précédents, le système ne peut déployer un processus de fusion que si une communauté d'agents immobiles émerge de la multiplicité d'agents mobiles, car seule une telle communauté active le système de contrôle. Ainsi, comme nous le présentons dans le Paragraphe 9.5.3, un agent mobile peut se transformer en agent immobile et réciproquement.

9.5.2 Agent immobile

Définition 9.5.2 (Agent immobile)**.** *Au contraire d'un agent mobile, un agent immobile est associé à des architectures plus puissantes et disposant d'une connexion réseau permanente. De là, un agent immobile est privilégié pour le contrôle du système et pour l'exécution de traitements gourmands en ressources CPU et mémoire.*

Après une analyse des réseaux composés d'architectures prédisposées à l'exécution d'agents immobiles, nous avons constaté qu'une communauté de tels agents formait une clique au niveau du graphe de connexion, par exemple un réseau domestique. Par extension, nous avons également constaté que plusieurs communautés pouvaient être définies sur des infrastructures plus importantes, comme par exemple un bâtiment public ou un atelier de fabrication.

Dans notre approche, nous proposons donc un contrôle du système basé sur des communautés d'agents immobiles où chaque communauté déploie un processus de fusion d'informations. Pour l'heure nous écartons les déploiements de processus de fusion sur plusieurs communautés ainsi que le déploiement de plusieurs processus concurrents au sein de la même communauté.

Dès lors que des agents mobiles se réunissent en une communauté d'agents immobiles, celle-ci suit les objectifs suivants :

- le déploiement d'un processus de fusion donné ;
- le contrôle de l'exécution du processus de fusion ;
- le contrôle du système réparti qui héberge tous les agents.

Ces trois objectifs sont atteints grâce aux tâches de contrôle présentent dans le système. Ainsi, le déploiement est basé sur les tâches *Configuration* et *Implémentation*, alors que les deux autres objectifs reposent sur les tâches des couches de contrôle supérieures (c.f. Figure 9.7).

Alors que les agents mobiles n'activent que le niveau le plus bas des tâches de contrôle, les agents immobiles peuvent activer l'unique niveau supérieur des tâches *Robustesse* et *Performance* ainsi que la hiérarchie des tâches *Configuration* et *Reconfiguration*. Cette activation d'un niveau de contrôle est basée sur une organisation autonome du système, présentée dans le Paragraphe 9.6.

9.5.3 Transformations

D'agent mobile à agent immobile La transformation d'un agent mobile en un agent immobile a lieu dans plusieurs cas. Le premier cas concerne l'intégration d'un agent mobile dans une communauté d'agents immobiles. En effet, dans un tel cas, la connexion entre l'agent mobile et au moins un des agents de la communauté est établie suite à la phase de découverte. La transformation de l'agent mobile a alors lieu si les deux conditions suivantes sont satisfaites :

- l'agent mobile dispose d'une connexion réseau permanente avec tous les agents de la communauté qu'il va intégrer ;
- l'agent mobile dispose d'une puissance de calcul suffisante pour exécuter des niveaux de contrôle supérieurs.

Si ces deux conditions sont vérifiées, l'agent mobile devient immobile et active des niveaux de contrôle supérieurs comme nous le présentons dans le Paragraphe 9.7. Après l'activation d'un des niveaux de contrôle, l'évolution de la position de l'agent dans la hiérarchie s'effectue par des mécanismes que nous présentons dans le Paragraphe 9.8.2.

L'autre cas de transformation traite de la naissance d'une communauté d'agents immobiles à partir d'agents mobiles. Cette naissance correspond à la création de l'état initial du système complexe créé à partir des différentes tâches de contrôle en interaction. Plusieurs mécanismes sont possibles, comme par exemple des approches basées sur une phase de négociation entre les agents mobiles. Ceux-ci doivent alors définir quelle est l'attribution de chaque tâche de contrôle.

À l'heure actuelle nous réalisons cette étape par une initialisation de l'agent immobile au niveau le plus bas. En outre, le niveau de contrôle activé par un agent est le même pour toutes les tâches du système.

Dès lors que l'état initial du système de contrôle est défini, l'évolution de la hiérarchie entre agents s'effectue selon le mécanisme que nous décrivons dans le Paragraphe 9.7.

D'agent immobile à agent mobile Si le passage de mobile à immobile fait intervenir plusieurs étapes à franchir par un agent, la transformation inverse est au contraire assez simple. En effet, celle-ci a lieu dès qu'une des conditions nécessaires à l'existence d'un agent immobile n'est plus satisfaite.

C'est par exemple le cas quand un agent immobile perd assez de puissance pour ne plus être en mesure d'accomplir une des tâches de contrôle qui lui étaient affectées. Dès qu'un agent immobile ne peut plus exécuter au moins un des niveaux supérieurs d'une tâche de contrôle, il redevient mobile.

L'autre contrainte concerne la connexion réseau d'un agent immobile, noté *agent1*. Si celle-ci est intermittente avec un des agents de la communauté, noté *agent2*, alors *agent1* devient mobile. *Agent2* ne devient pour autant pas automatiquement mobile. En effet, celui-ci subit cette transformation uniquement s'il est dans exactement la même situation que *agent1* : i.e. pour le même nombre de connexions réseau perdues et pour la même puissance. Cette hypothèse nous préserve des connexions répétées de plusieurs duos d'agents. Ainsi, si par exemple deux agents immobiles rompent leur connexion réseau, mais conservent celle avec le reste de la communauté, alors le plus puissant des deux restera dans la communauté d'agents immobiles.

Afin de garantir la cohérence du système de contrôle, des mécanismes de réorganisations sont mis en œuvre lors du passage d' un agent immobile à mobile. Nous présentons ces mécanismes de transition dans le Paragraphe 9.8.3.

9.6 Mécanisme de différentiation et d'auto-stabilisation

Notre mécanisme d'auto-organisation de la hiérarchie de contrôle s'inspire du modèle d'auto-stabilisation présenté dans [LGMK05]. Dans cet article, les auteurs présentent deux modèles d'expression de gènes et de différentiation cellulaire basés sur la concentration de régulateurs dans les tissus biologiques. Nous ne détaillerons pas ce domaine bien loin du nôtre dans ce mémoire. En effet, nous nous focalisons sur les modèles présentés et le comportement émergent sous-jacent. La suite de ce paragraphe présente ces deux modèles puis nous expliquons le lien que nous avons établi entre le mécanisme d'auto-différentiation cellulaire et d'auto-organisation hiérarchique.

9.6.1 Modèle 1

Ce modèle définit une population de cellules de deux types A et B, comme l'illustre la Figure 9.10. Chaque cellule produit une quantité aléatoire R_a et R_b de molécule de type a ou b selon le type de la cellule, sur un intervalle de temps fixé (cet intervalle correspond au pas de simulation dans l'approche proposée). Chaque molécule se diffuse dans le système et le taux de consommation R_d fournit la quantité de molécule capturée par les cellules sur un intervalle de temps fixé.

Si l'on pose n la concentration locale d'une des molécules et D un coefficient de diffusion, alors la diffusion d'une molécule est $\frac{\delta n}{\delta t} = D\Delta n$. Si initialement aucune molécule n'est diffusée, après un intervalle de temps t le nombre de molécules à une distance r de sa source est

$$N(r,t) = \frac{N_0}{4\pi Dt} e^{-\frac{r^2}{4Dt}}$$

Une fois le nombre de molécules défini, le premier modèle présenté propose le calcul de la probabilité qu'une cellule change de type. Cette probabilité, notée P, est une fonction de la concentration des molécules a et b et deux interactions sont alors définies. La première considère que la concentration d'une molécule va stabiliser l'état des cellules du même type ; ce que les auteurs nomment auto-stabilisation. La seconde pose que la concentration d'une molécule stabilise au contraire l'autre type de cellule dans une inter-stabilisation. Pour un type donné T et son opposé T^-, la probabilité de changement pour une cellule est

$$P(T \rightarrow T^-) = q_{auto}F(NT) + q_{inter}F(NT^-)$$

avec NT et NT^- la quantité de molécules présentes dans l'environnement d'une cellule. Le modèle présenté base le calcul de cette probabilité sur la fonction Fermi-Dirac suivante :

$$F(x) = \frac{1 + e^{-\beta C_0}}{1 + e^{\beta(x - C_0)}}$$

Les détails de cette fonction sortent du cadre de notre approche et nous renvoyons à l'article original pour une plus ample description. Les auteurs utilisent ce premier modèle pour une analyse des effets de l'auto- et de l'inter-stabilisation. Néanmoins, aucune structure n'en émerge et un autre modèle est alors proposé.

9.6.2 Modèle 2

Le second modèle présenté propose une auto-stabilisation uniquement. Néanmoins, une cellule d'un type donné a besoin de la molécule produite par l'autre type de cellule pour proliférer. De là découle une interdépendance de la prolifération.

La survie d'une cellule dépend donc de la quantité de molécule qu'elle va consommer. Celle-ci est établie par un tirage aléatoire sur une distribution Gaussienne de probabilités. Si la quantité consommée sur un intervalle de temps C_c est supérieure à celle présente dans le voisinage de la cellule, alors celle-ci meurt.

L'augmentation de la population d'un type de cellule peut être réalisée de deux façons : soit une cellule change de type, soit deux cellules sont créées à partir d'une cellule (par mitose). Le second cas de prolifération se produit lorsque que la concentration d'une molécule dans un voisinage est suffisamment important pour que des cellules supplémentaires de l'autre type puissent survivre.

À l'issue de la présentation de ces deux modèles, les auteurs soulignent l'importance de l'état initial du système. Alors que dans la nature, des cellules spéciales stockent une quantité suffisante des deux types de molécule pour initialiser le processus, notre approche propose une autre solution comme nous le présentons dans le paragraphe suivant.

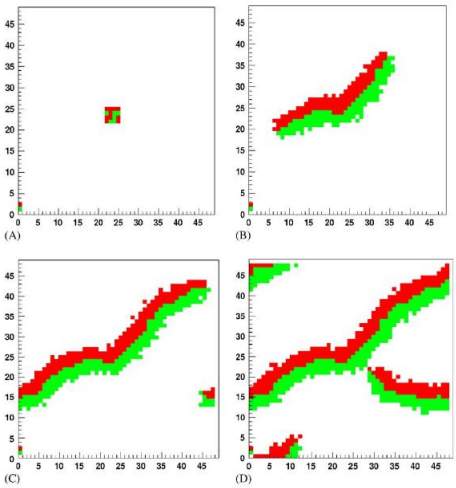


FIG. 9.10 – Résultat de la simulation d'automates cellulaires auto-stabilisés tiré de [LGMK05].

9.7 Activation et désactivation d'un niveau de contrôle

En nous inspirant des modèles présentés dans le Paragraphe 9.6, nous proposons un mécanisme d'auto-organisation de la hiérarchie d'une tâche de contrôle. Chaque cellule du modèle initial est un agent immobile de notre approche et l'état d'une cellule correspond au niveau de contrôle activé par l'agent. L'idée est d'étendre le mécanisme proposé dans [LGMK05] :

- à plusieurs types de cellules ;
- en relâchant la notion de distance entre les cellules.

La notion de « type » de cellules est traduite dans notre approche par le « niveau » de contrôle activé par un agent immobile et, si l'on considère le nombre de connexions réseau entre deux agents, tous les agents immobiles du système sont équidistants par construction.

9.7.1 Interaction de contrôle

Il paraît clair que nos agents logiciels ne vont pas émettre de molécules dans le système. Ainsi, nous proposons de traduire la production d'une molécule par l'émission d'un message de contrôle. Ce message est émis d'un niveau de contrôle vers les niveaux adjacents et la fréquence de ces messages représente la concentration d'une molécule. Le résultat, illustré par la Figure 9.11, est donc une différentiation des agents et une auto-organisation sur plusieurs niveaux.

Orientation de l'organisation Notre approche étend l'auto-organisation du système en orientant la construction de la hiérarchie d'une tâche de contrôle. En effet, les agents qui composent le système de contrôle ont des caractéristiques techniques différentes, notamment au niveau de leur puissance de calcul.

De là, nous proposons qu'une relation existe entre la puissance d'un agent donné, la taille de la population d'agents qu'il contrôle et la fréquence des messages de contrôle envoyés. En conséquence, nous avons analysé cette relation et nous avons exhibé des plages de valeurs pour les paramètres de notre mécanisme de répartition. L'analyse de ces paramètres est basée sur les résultats de simulation, effectuées sur plusieurs types de configurations, et regroupés dans le Chapitre 10.

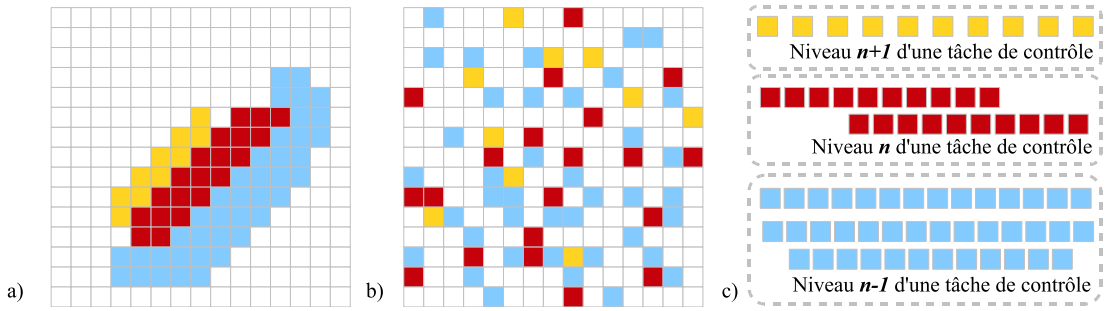


FIG. 9.11 – Mécanisme de différentiation et d'auto-stabilisation appliqué à l'organisation d'une tâche de contrôle hiérarchique : a) trois groupes d'agents pour trois niveaux de contrôle adjacents, b) vue du système sans la notion de distance, et c) hiérarchie de contrôle basée sur les groupes d'agents.

9.7.2 Limites d'activation

Afin de compléter l'adaptation des modèles présentés dans le Paragraphe 9.6, nous présentons ci-dessous la traduction des concentrations de molécule nécessaires à la survie, à la stabilisation et à la prolifération d'une cellule.

Dans notre modèle, nous posons qu'une cellule est vivante si l'agent qu'elle traduit émet des messages de contrôle. Ainsi, le cas particulier du passage d'un agent mobile à un agent immobile (c.f. Paragraphe 9.5.3) correspond à la naissance d'une cellule.

Le changement de niveau de contrôle correspond à la mort de la cellule qui quitte un niveau et à la naissance de celle-ci dans le niveau adjacent. La traduction du cas de prolifération par mitose n'a pas réellement de sens dans notre approche car un agent ne peut pas subir de « division ».

Concentrations limites Notre approche propose la gestion d'un système de fusion en cours d'exécution et, à la différence de l'article sur lequel nous nous basons, les concentrations limites de survie et de stabilisation ne seront pas simulées mais mesurées sur le système. Ainsi, pour un intervalle de temps fixé, chaque agent établit la fréquence des messages de contrôle émis par le niveaux adjacents et par son niveau de contrôle : respectivement F_{n-1},F_{n+1} et F_n. Les messages émis par les agents d'un niveau donné stabilisent les agents de ce niveau et « nourrissent » les agents des niveaux intermédiaires. En effet, nous définissons deux fréquences « de survie », notées F_u et F_d, qui correspondent aux fréquences de contrôle nécessaire à un agent du niveau n, respectivement depuis les niveaux $n+1$ pour F_u et $n-1$ pour F_d.

Pour un agent de niveau n donné, si $F_{n-1} < F_d$ alors l'agent descend d'un niveau de contrôle, et si $F_{n+1} < F_u$ alors l'agent monte d'un niveau de contrôle. Ces deux cas correspondent à un déséquilibre entre deux niveaux de contrôle. Ainsi, la montée d'un agent augmente la puissance de calcul du niveau supérieur tout en réduisant la charge de contrôle nécessaire à la gestion de son niveau initial.

À coté des tendances liées aux messages de contrôle issus des niveaux adjacents, un agent a également une certaine probabilité de monter d'un niveau selon la valeur de F_n. Ainsi, la probabilité qu'un agent monte d'un niveau sera :

$$P(n \to n+1) = \frac{1 + e^{-\beta C_0}}{1 + e^{\beta(F_n - C_0)}}$$

9.8 Transition inter-niveaux

9.8.1 Niveaux activables

Notre approche base le contrôle hiérarchique du système sur une organisation autonome des agents qui le composent. Ces agents activent deux niveaux de la hiérarchie pour chaque tâche de contrôle. Le niveau de contrôle le plus bas est activé par tous les agents mobiles et immobiles pour permettre le déploiement d'une partie de la configuration. Un niveau supérieur n'est activé que sur les agents immobiles car seuls eux disposent de suffisamment de ressources pour effectuer le contrôle du système. Le choix du second niveau activé est basé sur le mécanisme décrit dans le Paragraphe 9.7.

9.8.2 Mécanismes d'activation

Dans le but de garantir une certaine cohérence dans le contrôle du système, nous proposons trois types de mécanismes d'activation d'un niveau de contrôle. Ainsi, quand un agent active un niveau de la hiérarchie, les agents du même niveau agissent comme suit :

- pas de coordination entre l'agent et les agents du niveau activé ;
- ou une division de l'espace de contrôle, i.e. de la population d'agents gérés par un niveau de contrôle ;
- ou une synchronisation des requêtes de contrôle entre l'agent et les agents du niveau activé.

Si l'absence de coordination est intuitive, les deux modes de coordination nécessitent quelques éclaircissements.

La division de l'espace de contrôle, illustrée par la Figure 9.12, consiste en une nouvelle attribution des relations de contrôle entre les agents d'un niveau et ceux du niveau inférieur. Ainsi une partie des groupes gérés par les agents initialement présents à un niveau de contrôle est affectée à l'agent arrivant.

La synchronisation des requêtes de contrôle, illustrée par la Figure 9.13, a également pour but la diminution de la charge de contrôle par agent d'un niveau. Dans le cadre de cette coordination, la réduction de la charge s'effectue sur la fréquence des requêtes de contrôle. Les agents d'un même niveau effectue alors un contrôle à tour de rôle sur les agents du niveau inférieur.

Seul la coordination par division de l'espace de contrôle a été retenue dans notre approche. En effet, la synchronisation des requêtes de contrôle impliquait la mise en place, au moins partielle, d'une horloge logique globale. Or ce type de mécanisme est contraire à l'idée qui guide notre approche, où nous souhaitons baser le comportement d'un agent uniquement sur des informations locales.

9.8.3 Mécanismes de désactivation

Toujours dans un souci de cohérence du système de contrôle, le départ d'un agent d'un niveau de la hiérarchie s'effectue comme suit :

- pas de coordination ;
- ou une nouvelle affectation des agents contrôlés ;
- ou une nouvelle affectation du groupe d'agents contrôlés.

Là encore, une absence de coordination est intuitive : dans ce cas, un agent qui perd l'agent chargé de son contrôle est à l'initiative d'une réintégration dans la hiérarchie. Cette étape de réintégration reste nécessaire lorsqu'un agent du système de contrôle subit une défaillance.

Dans le cas d'une désactivation planifiée, les deux coordinations que nous proposons ici ont pour but le maintien du contrôle pendant l'évolution de la hiérarchie. La première coordination partage le groupe d'agents contrôlés entre les agents du niveau quitté, alors que la seconde affecte le contrôle du groupe d'agents contrôlés à un seul agent du niveau quitté.

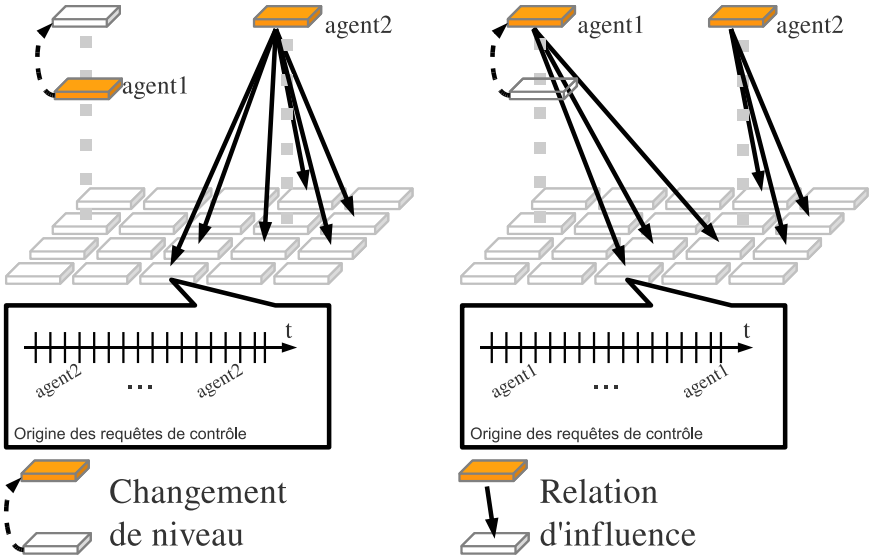


FIG. 9.12 – Activation d'une niveau de la hiérarchie de contrôle : coordination par division du contrôle.

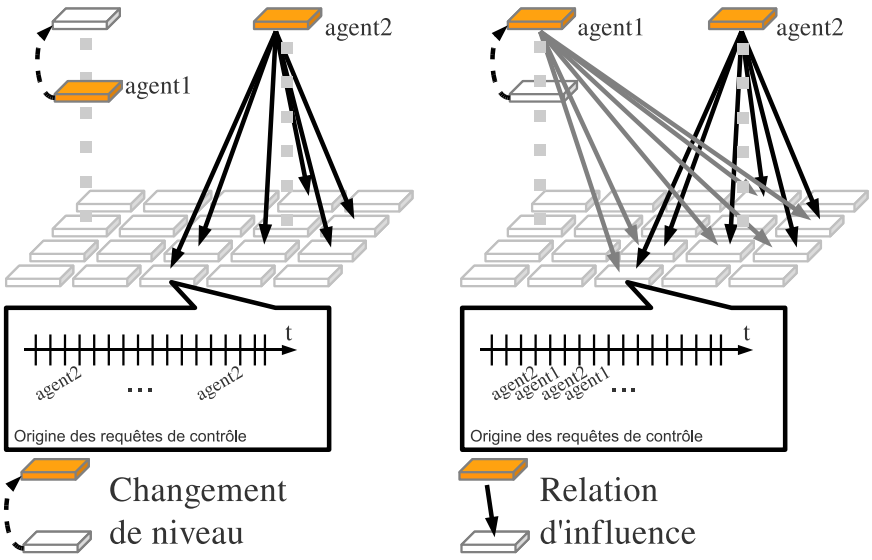


FIG. 9.13 – Activation d'une niveau de la hiérarchie de contrôle : coordination par synchronisation du contrôle.

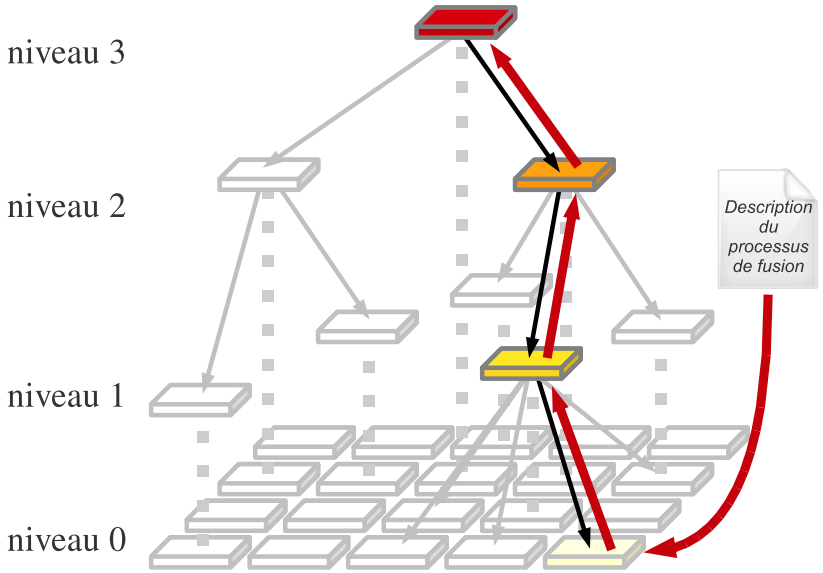


FIG. 9.14 – Communication au sein d'une tâche de contrôle.

Notre approche a retenu ces deux coordinations, sélectionnées selon un critère de désactivation. Ainsi, si la désactivation doit être effectuée au plus tôt, une nouvelle affectation d'un groupe d'agents dans son ensemble est plus rapide à effectuer. La correction de l'éventuel déséquilibre induit par cette coordination est alors laissée à la charge des agents restants. La seconde coordination, plus longue à effectuer, nécessite une coopération de tous les agents du niveau de contrôle.

9.9 Communication de contrôle

9.9.1 Au sein d'une tâche de contrôle

Une communication de contrôle est définie comme un message transmis entre les niveaux d'une tâche, comme l'illustre la Figure 9.14, et qui nécessite un traitement ou une action du destinataire. Ce type de communication est indépendant des interactions de contrôle définies dans le Paragraphe 9.7.1 et s'effectue parallèlement dans la hiérarchie.

Ainsi, un message émis par le bas de la hiérarchie à destination des niveaux supérieurs est identifié comme une requête qu'un des agents de la tâche de contrôle doit traiter. Au contraire, un message transmis du haut vers le bas de la hiérarchie est identifié comme une commande qu'un agent doit effectuer.

La transmission des messages est assez simple et suit les deux règles suivantes :

- si le message est ascendant et que l'agent qui le réceptionne peut traiter la requête alors il le fait et envoie éventuellement un message descendant en réponse. S'il ne peut pas traiter la requête, l'agent la transmet au niveau supérieur s'il existe, sinon une erreur est générée ;
- si le message est descendant et que l'agent qui le réceptionne en est le destinataire, alors il effectue l'action décidée par le système de contrôle, sinon il transmet le message au niveau inférieur s'il existe.

9.9.2 Entre deux tâches de contrôle

Une communication entre deux tâches de contrôle est également très simple. Elles sont toujours liées à une requête par opposition aux messages descendants du paragraphe précédent. Le sens de ces communication suit l'organisation des tâches de contrôle déjà présentées et illustrées par la Figure 9.7.

Rappelons que tous les agents du système activent le niveau le plus bas de chaque tâche de contrôle, comme l'illustre la Figure 9.9. Chaque agent appartient donc à la hiérarchie de toutes les tâches de contrôle. Cela permet donc, au sein d'un agent, la communication directe d'une tâche de contrôle vers une autre tâche. Chaque message reçu par le niveau le plus bas d'une hiérarchie donne lieu à un nouveau message interne à une tâche de contrôle.

9.10 Synthèse

Ce chapitre présente le dernier apport théorique de ce mémoire. L'originalité de notre mécanisme de répartition découle de l'auto-organisation de la hiérarchie de contrôle. L'approche bio-inspirée décrite dans ce chapitre permet l'adaptation de la charge de contrôle affectée à chaque agent du système en fonction de la taille de la population d'agents à gérer, ainsi que leur puissance de calcul.

Cette adaptation a en l'occurrence lieu lors de l'arrivée d'agents dans le système, en donnant lieu à une croissance en largeur de la hiérarchie de contrôle. La hauteur de la structure de contrôle s'adapte au contraire en fonction de la puissance de calcul des agents. En effet, alors que le mécanisme d'origine définit un comportement du système selon deux types de cellules, notre approche propose que pour un agent considéré, trois choix s'offrent à l'agent : rejoindre un des deux niveaux adjacents, ou rester à son niveau dans la hiérarchie de contrôle. Si le niveau d'un agent doit être adapté, par exemple en raison de sa puissance de calcul, nous définissons un coefficient qui oriente le mouvement d'un agent. Une propriété structurelle émerge alors de la hiérarchie de contrôle et l'on voit apparaître une hausse de la puissance de calcul moyenne des agents du bas de la hiérarchie vers le haut.

Si le modèle de répartition est clairement énoncé, certains points sont à préciser, notamment l'importance de chaque paramètre du modèle, ainsi que leur valeur selon les caractéristiques de la communauté d'agents.

Chapitre 10

Analyse du mécanisme de répartition

À présent que nous avons présenté notre modèle de répartition du système de contrôle, nous proposons dans ce chapitre l'analyse menée sur la simulation de ce modèle. En effet, le système que nous utilisons pour tester les différents points de cette thèse ne nous permet actuellement pas d'évaluer notre mécanisme de répartition sur un cas réel. Nous avons donc choisi de le modéliser dans le cadre formel DEVS et de le simuler sur plusieurs plages de configurations possibles.

Ce chapitre présente dans un premier temps le cadre de modélisation que nous utilisons dans la suite pour décrire notre mécanisme de répartition. La mise en œuvre de l'expérimentation est ensuite présentée puis nous terminons ce chapitre par nos résultats.

10.1 Modèle DEVS

La vérification du mécanisme de répartition que nous proposons dans ce chapitre, repose sur une spécification du système dans un modèle d'analyse. Le système peut alors être décrit à plusieurs niveaux, comme les niveaux de connaissance relative au système, ou encore selon la structure abstraite manipulée [ZPK00]. Ce dernier modèle de spécification propose cinq niveaux.

Niveau 0 Ce niveau, appelé interface d'observation ou *Observation Frame* (OF), définit quelles sont les variables à mesurer et comment les observer. Il est formellement défini par le tuple

$$M_{OF} =< X, Y, T > \tag{10.1}$$

tel que :
- X est l'ensemble des entrées du système ;
- Y est l'ensemble des sorties du système ;
- T est la base temporelle d'expression du système.

Niveau 1 Dans le niveau de base de données comportementale, ou *Input Output Relation Observation* (IORO), le comportement du modèle est fondé sur un ensemble de données collectées à partir du système d'origine. Un résultat du système est donc issu d'une relation dont les ensembles de valeurs sont définis par un historique de mesures. Ces deux ensembles peuvent en outre être discrets ou continus selon la granularité des mesures fournies. Ce type de niveau de spécification est donc défini par le tuple

$$M_{IORO} =< X, Y, T, \Omega, R > \tag{10.2}$$

tel que :
- X, Y, T comme définis précédemment ;

- Ω est l'ensemble des données possibles avec $\Omega \subset X \times T$;
- $R = \{(\omega, \rho) | dom(\omega) = dom(\rho)\}$ est la base de données comportementales avec $R \subset \Omega \times Y \times T$.

Niveau 2 Ce type de spécification est basé sur un ensemble de fonctions comportementales du système. Ainsi, à la différence du niveau précédent, le modèle du système permet de calculer un résultat pour chaque entrée du système. Un modèle décrit par ce type de spécification, aussi appelée *Input Output Function Observation* (IOFO), est défini par un tuple de la forme

$$M_{IOFO} =< X, Y, T, \Omega, F > \tag{10.3}$$

tel que :

- X, Y, T, Ω comme définis précédemment ;
- $f \in F$ une fonction comportementale avec $\rho = f(\omega)$ et $dom(\omega) = dom(\rho)$.

Niveau 3 Ce niveau de spécification base la description du modèle du système sur un ensemble de transitions d'état. Ainsi, là où les niveaux précédents ne faisaient intervenir que les entrées et les sorties du système, ce niveau décrit le comportement interne du système. Un modèle de ce niveau de spécification, encore appelé *Input Output System* (IOS), est décrit par un tuple

$$M_{IOS} =< X, Y, T, S, \delta_{ext}, \delta_{int}, \lambda, t_a > \tag{10.4}$$

tel que :

- X, Y, T, comme définis précédemment ;
- S est l'ensemble des états internes du système ;
- $\delta_{ext} : Q \times X \rightarrow S$ est la fonction de transition externe avec
  - $Q = \{(s, e) | s \in S, 0 \leq e \leq t_a(s)\}$ l'ensemble des états possibles du système dans le temps ;
- $\delta_{int} : S \rightarrow S$ est la fonction de transition interne ;
- $\lambda : S \rightarrow Y$ définit la fonction de sortie ;
- $t_a : S \rightarrow T \cup \infty$ est la fonction d'avance temporelle.

Un système décrit à ce niveau de spécification est dans un état $s \in S$ à un instant donné et doit y rester pour un délai $t_a(s)$ donné si aucun événement extérieur ne vient le perturber.

Lorsque le délai $e = t_a(s)$ (*elapsed time*) est écoulé sans qu'aucun événement extérieur ne soit survenu, le modèle décrit l'état suivant $y = \lambda(s)$, et effectue une transition d'état interne de l'état s vers $\delta_{int}(s)$. Si au contraire un événement extérieur $x \in X$ survient avant l'expiration du délai e, le modèle décrit une transition d'état externe de s vers $\delta_{ext}(s, e, x)$. Dans les deux cas, le système se retrouve alors dans un nouvel état s', pour un délai définit par $t_a(s')$.

Niveau 4 Ce dernier niveau de spécification permet la description du système sous la forme d'une composition de modèles. Dès lors, il est donc non seulement possible de combiner des modèles issus de différentes spécifications, mais également de créer une hiérarchie de modèles. Les modèles interconnectés sont donc regroupés dans un modèle de type *Coupled Network* (CN), défini par le tuple

$$M_{CN} =< X, Y, T, D, \{M_d\}, \{I_d\}, Z_{i,d}, Select > \tag{10.5}$$

tel que :

- X, Y, T, comme définis précédemment ;
- D contient le nom des composants du modèle couplé ;

- M_d est le modèle du composant $d \in D$;
- I_d est l'ensemble des influenceurs de d, i.e. des composants qui produisent un message pour d, avec CN le modèle couplé et $I_d \subseteq D \cup \{CN\}, d \neq I_d$;
- $Z_{i,d}$ est une fonction de transfert de messages avec
  - $Z_{CN,d} : X \to X_d$;
  - $Z_{i,CN} : Y_i \to Y$;
  - $Z_{i,d} : Y_i \to X_d, i \neq CN, d \neq CN$;
- $Select : 2^D \to D$ est une fonction d'arbitrage nécessaire à l'ordonnancement d'événements synchrones.

10.2 Modélisation des agents

Nous avons choisi de modéliser un agent du système de contrôle par un modèle de type *Input Output System*, ce qui nous permet de décrire les interactions qu'il a avec son entourage et ainsi déterminer son état interne. La suite de ce paragraphe présente ce modèle et nous détaillons les fonctions de transitions externe et interne du système à la fin de ce paragraphe.

Modèle d'un agent

Un agent est modélisé par un tuple

$$M_{agent} =< X, Y, T, S, \delta_{ext}, \delta_{int}, \lambda, t_a > \tag{10.6}$$

tel que :

- $X = Y = \{\text{plop}_l\}$ est le singleton qui représente le message échangé entre deux agents avec $l \in \mathbb{N}$ le niveau de l'agent qui a émis le message ;
- $T = \mathbb{R}$: nous avons choisi de modéliser un agent sur une base de temps réelle ;
- $S = \{(K, \sigma, l, H, w, U, t)\}$ est l'ensemble des états internes d'un agent tel que :
  - $K \in T$ est le temps de traitement d'un message par l'agent ;
  - $\sigma \in T$ est le temps avant l'envoi du prochain message ;
  - $l \in \mathbb{N}$ représente le niveau de l'agent dans la hiérarchie de contrôle ;
  - $H \in \mathbb{R}^3$ est un triplet de réels qui représente le nombre de messages émis par le niveau inférieur dans $H[0]$, le niveau de l'agent dans $H[1]$ et le niveau supérieur dans $H[2]$;
  - $w \in \mathbb{N}$ est un compteur qui inhibe un changement de niveau trop proche d'un précédent changement quand sa valeur est positive ;
  - $U \in T^3$ est un triplet de réels qui représente l'intervalle de temps depuis la dernière mise à jour de H ;
  - $t \in T$ est une horloge locale ;
- $\delta_{ext} : Q \times X \to S$ est la fonction de transition externe que nous détaillons dans la suite de ce paragraphe ;
- $\delta_{int} : S \to S$ est la fonction de transition interne également détaillée dans la suite de ce paragraphe ;
- $\lambda(s) = \text{plop}_l$ est la fonction de production d'un message à partir d'un état de l'agent ;
- $t_a(s) = \sigma$ est la fonction d'avance temporelle.

Fonction de transition externe

Pour compléter la spécification du modèle d'un agent, voici en détail la fonction de transition externe définie par $\delta_{ext}(s, e, x) = s'$ avec :

- $s = (K, \sigma, l, H, w, U, t)$ est l'état courant de l'agent ;

- $e \in T$ est le temps écoulé depuis la dernière transition ;
- $x = \text{plop}_{l''}$ est le message reçu ;
- $s' = (K', \sigma', l', H', w', U', t')$ est l'état de l'agent à l'issue de la transition.

Pour simplifier la lecture de la description de cette fonction, nous supposons que toutes les composantes non explicitement modifiées sont inchangées entre l'état initial s et l'état suivant s'.

Pour tout état s Quelque soit l'état initial de l'agent, notre modèle définit la mise à jour du temps restant avant le prochain envoi d'un message et la mise à jour du nombre de *plop* reçus suivant le niveau d'émission de x :

$$\forall l \left| \begin{array}{l} \sigma' = \sigma - e \\ t' = t + e \end{array} \right. \tag{10.7}$$

$$l'' < l \left| \begin{array}{l} H'[0] = \Delta(H[0], t - U[0]) \\ U'[0] = t \end{array} \right. \tag{10.8}$$

$$l'' = l \left| \begin{array}{l} H'[1] = \Delta(H[1], t - U[1]) \\ U'[1] = t \end{array} \right. \tag{10.9}$$

$$l'' > l \left| \begin{array}{l} H'[2] = \Delta(H[2], t - U[2]) \\ U'[2] = t \end{array} \right. \tag{10.10}$$

Mouvement autorisé La suite de cette présentation introduit les paramètres du modèle de simulation suivants :

- q_{auto} et q_{inter} modélisent la part d'auto-stabilisation et d'inter-stabilisation dans le mécanisme de différenciation ;
- q_{level} oriente le mouvement des agents et le sens de la hiérarchie de contrôle ;
- K_w est une constante qui modélise le temps d'inhibition du mouvement suite à un changement de niveau.

Nous donnons la liste complète des paramètres du modèle dans le Paragraphe 10.3.2. La modification du niveau de l'agent est différente selon l'inhibition du changement de niveau. Dans le cas où le changement de niveau n'est pas inhibé, i.e. $w = 0$, nous avons :

$$\forall l \left| \; p_{mv} = q_{auto} * F(H[1]) + q_{inter} * F(H[0] + H[2]) \right. \tag{10.11}$$

$$p > p_{mv} \left| \begin{array}{l} l = 1 \vee H[0] > H[1] * q_{level} \vee H[1] > H[2] * q_{level} \Rightarrow l' = l + 1 \\ l > 1 \wedge H[0] < H[1] \Rightarrow l' = l - 1 \end{array} \right. \tag{10.12}$$

$$p < p_{mv} \left| \; l' = l \right. \tag{10.13}$$

$$l \neq l' \Rightarrow w' = K_w \tag{10.14}$$

Mouvement inhibé Dans le cas où le changement de niveau est inhibé, i.e. $w > 0$, le modèle décrit uniquement la mise à jour de w avec :

$$\forall l, w' = w - 1 \tag{10.15}$$

Fonction de transition interne

Pour terminer la description du modèle d'un agent, voici les détails de la fonction de transition interne définie par $\delta_{int}(s) = s'$ avec :

- $s = (K, \sigma, l, H, w, U, t)$ est l'état interne initial de l'agent ;
- $s' = (K', \sigma', l', H', w', U', t')$ est l'état interne de l'agent à l'issue de la transition.

Ici encore nous supposons que toutes les composantes non explicitement modifiées sont inchangées entre l'état initial s et l'état final s'.

Pour tout s Quelque soit l'état interne de l'agent, le modèle décrit la mise à jour du temps restant avant le prochain envoi de message tel que :

$$\begin{aligned} t' &= t + \sigma \\ \sigma' &= K \end{aligned} \tag{10.16}$$

Mouvement autorisé La fonction de transition interne définit elle aussi deux types d'états finaux selon l'inhibition des mouvements entre niveaux. Dans le cas où $w = 0$, le changement de niveau n'est pas inhibé est le modèle décrit alors la mise à jour du niveau de l'agent comme indiqué dans les Équations 10.11 à 10.14.

Mouvement inhibé Dans le cas où le mouvement est inhibé, $w > 0$, seule la valeur de w est mise à jour comme indiqué dans l'équation 10.15

Modèle d'un système

Nous avons choisi de modéliser la communauté d'agents à l'aide du niveau 4 de spécification que nous présentons au début de ce chapitre dans le Paragraphe 10.1. Ce type de spécification nous permet de décrire le modèle du système de contrôle comme la composition des modèles des agents en interactions.

Le système est donc modélisé par le tuple

$$M_{sys} =< X, Y, T, D, \{M_d\}, \{I_d\}, \{Z_{i,d}\}, Select >$$

tel que :

- $X = Y = \emptyset$ car le système de contrôle n'a pas d'interaction avec l'extérieur, il est donc inutile de définir des ensembles de messages échangés ;
- $T = \mathbb{R}$: nous avons choisi une base de temps réelle pour la simulation du système ;
- $D = \{A_i\}_{i \in [1..n]}$ est l'ensemble des noms des n agents du système ;
- $M_d = \{M_{A_i} | A_i \in D, M_{A_i} = M_{agent}\}$ comporte n fois le même modèle défini dans l'équation 10.6, utilisé par les agents du système ;
- $I_d = D/d$ est l'ensemble des influenceurs d'un agent $d \in D$;
- $Z_{i,d}$ est la fonction d'échange de messages, dans notre cas de diffusion, avec sys le modèle du système :
  - $Z_{i,d} : Y_i \rightarrow X_d, i \neq sys, \forall d \in I_d, d \neq sys$;
- $Select : 2^D \rightarrow D$ est la fonction d'arbitrage qui sélectionne aléatoirement un des événements synchrones.

10.3 Observations

La simulation du modèle DEVS de notre système de contrôle a été menée sur un simulateur implémenté en Java, développé à Clermont-Ferrand. Ce simulateur est couplé à une interface

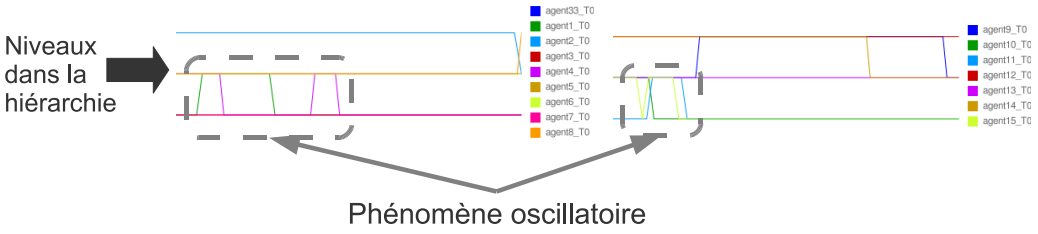


FIG. 10.1 – Observation de phénomènes oscillatoires.

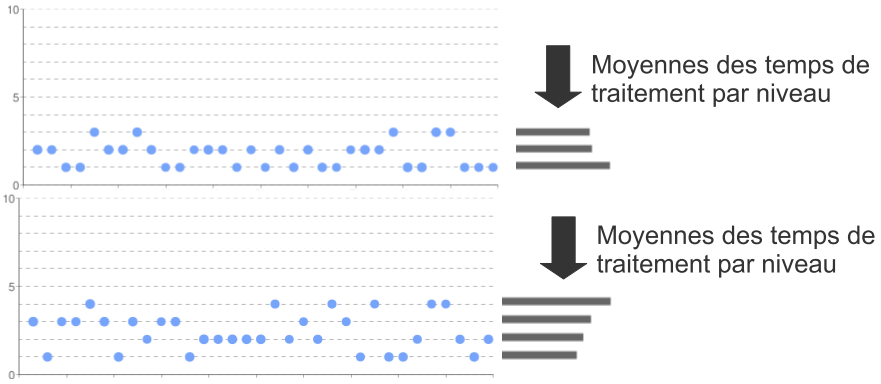


FIG. 10.2 – Observation de la répartition.

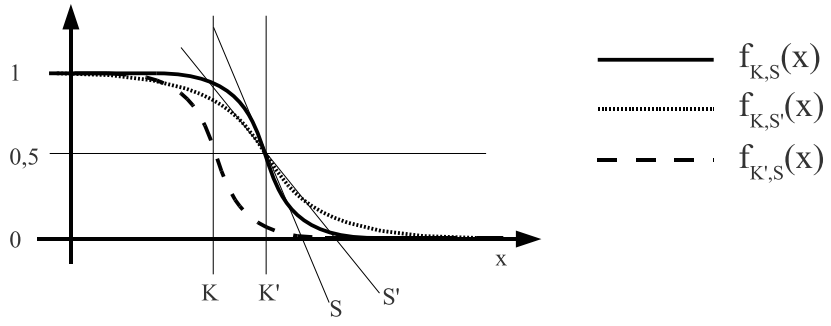


FIG. 10.3 – Profils de la fonction de probabilité de Fermi-Dirac : variation du point d'inflexion et de la pente.

graphique que nous avons développée, afin de visualiser le comportement de notre système ainsi que pour agir sur les attributs du modèle. Les performances du simulateur ne sont pas abordées dans ce mémoire et nous nous intéressons à deux éléments de l'interface graphique, repris dans les Figures 10.1 et 10.2. Le premier graphique présente l'historique des niveaux de contrôle occupés par les agents du système : la configuration illustrée présente une hiérarchie de contrôle sur trois niveaux. Le second graphique fournit une vue instantanée de la hiérarchie du système de contrôle, avec le calcul de la puissance moyenne des agents pour un niveau de contrôle. La Figure 10.2 présente la vue de deux configurations, respectivement sur trois et quatre niveaux de contrôle. La longueur des rectangles sur le côté droit évolue en fonction de la puissance moyenne des agents d'un niveau de contrôle.

Par la suite, cette interface graphique a été remplacée par une exportation des résultats d'une exécution dans un fichier CSV afin de faciliter l'automatisation de la campagne de simulation (plus de 300000 *runs*) et de l'exploitation des résultats qui représentent plus de 800 Mo de données CSV.

10.3.1 Premières hypothèses

Lors des première simulations, certains phénomènes ont été observés. En effet, des phénomènes oscillatoires, illustrés par la Figure 10.1, sont rapidement apparus. De plus, l'organisation de la hiérarchie de contrôle, illustrée par la Figure 10.2, était sensible à la valeur de certains paramètres conformément au modèle.

Sur la base des premières observations de simulation, nous avons formulé plusieurs propositions en fonction des valeurs de ces paramètres : la Figure 10.3 illustre deux des paramètres qui déterminent la probabilité de mouvement d'un agent, le Paragraphe 10.3.2 les reprend en détail.

Proposition 10.3.1 (Auto-stabilisation). *L'apparition des phénomènes oscillatoires est liée au choix du type de stabilisation.*

Cette proposition permet la comparaison de notre modèle de répartition et du modèle d'origine, présenté dans le Paragraphe 9.6. Les observations formulées à l'issue de l'analyse de cette proposition implique la formulation de propositions complémentaires nécessaire à l'analyse du comportement du système de répartition.

Proposition 10.3.2 (Influence de K). *Le point d'inflexion dans la fonction de Fermi-Dirac influe sur l'apparition de phénomènes oscillatoires.*

À la suite de l'analyse de la Proposition 10.3.1, nous concentrons notre réflexion sur la vérification de l'influence de K. En effet, conformément à la description du mécanisme de répartition, la valeur de K doit limiter les mouvements des agents dans le système.

Proposition 10.3.3 (Valeur de K). *Les valeurs du paramètre* K *limitant les phénomènes oscillatoires sont différentes selon le nombre et la puissance des agents dans le système.*

Le modèle du système est composé d'une communauté d'agents dont nous contrôlons la taille et l'homogénéité des puissances de calcul. L'analyse de cette proposition nous permet d'établir le comportement du mécanisme de répartition en fonction de ces deux paramètres.

Proposition 10.3.4 (Vivacité du système). *L'absence de phénomènes oscillatoires cache une stagnation de l'architecture de contrôle.*

Cette dernière proposition, énoncé intuitivement une vérification préalable à tout constat concernant l'influence des paramètres du modèle sur l'apparition d'une structure de contrôle stable.

10.3.2 Cadre expérimental

Les expérimentations que nous présentons dans ce paragraphe sont menées pour vérifier les propositions énoncées dans le Paragraphe 10.3.1. Pour cela, nous avons modélisé notre système en DEVS, conformément à la description que nous avons donnée dans le Paragraphe 10.2. La vérification est basée sur les résultats des exécutions où les paramètres du modèle varient comme suit - nous notons $p \in [a, b], \Delta = c$ le paramètre p défini sur l'intervalle $[a, b]$ avec un pas de simulation c :

- $nbAgent \in [5, 40], \Delta = 5$: ce paramètre représente le nombre d'agents dans le système ;
- $deltaPT \in [2, 5], \Delta = 1$: correspond à la différence maximale, du temps de traitement d'un message, pour l'ensemble des agents ; ce paramètre est utilisé pour le choix de la puissance d'un agent, dont la valeur est fixée selon un tirage aléatoire ;
- $qlevel \in [0.5, 3.0], \Delta = 0.25$: ce paramètre oriente le mouvement des agents, il est utilisé lors de la comparaison de l'interaction entre le niveau de l'agent en mouvement et le niveau supérieur ;
- $K \in [2.0, 20.0], \Delta = 0.5$: ce paramètre modélise la position du point d'inflexion dans la fonction de probabilités (c.f. Figure 10.3) ;
- $S \in [0.5, 2.0], \Delta = 0.5$: ce paramètre modélise la pente de la transition dans la fonction de probabilités (c.f. Figure 10.3) ;
- $qauto \in [0.5, 1.0], \Delta = 0.5$: modélise la part d'auto-stabilisation par rapport à l'inter-stabilisation lors du calcul de la probabilité de mouvement.

Pour chaque exécution de simulation, nous recueillons les mesures suivantes, pour chaque agent :

- $A \in \mathbb{N}$: est le niveau attracteur de l'agent. Nous le calculons comme le niveau le plus fréquenté dans la distribution des niveaux atteints par l'agent ;
- $Sec \in \mathbb{N}$: correspond au second niveau le plus fréquenté dans la distribution des niveaux atteints par l'agent ;
- $ratio \in \mathbb{R}$: est le rapport entre la valeur de A et Sec dans la distribution des niveaux atteints, cette mesure représente l'importance de l'attraction de A ;
- $deltaMax \in \mathbb{N}$: représente le nombre de niveaux différents atteints par l'agent ;
- $Fi0 \in \mathbb{N}$: est égale au nombre de changements de niveau effectués sur l'exécution ;
- $Fi1 \in \mathbb{N}$: est égale au nombre de changements de niveau effectués sur les 2000 dernières itérations ;
- $Fi2 \in \mathbb{N}$: est égale au nombre de changements de niveau effectués sur les 500 dernières itérations.

Auto-stabilisation

Analyse A Le but de cette analyse est de montrer l'influence du type de stabilisation du système. En effet, conformément au modèle dont nous nous sommes inspirés, notre mécanisme d'auto-organisation définit un paramètre, *qauto*, nous permettant de fixer la part de l'auto-stabilisation du système.

En effet, deux types de stabilisations sont possibles selon qu'un agent est influencé par les agents du même niveau de contrôle ou par les agents des niveaux adjacents. Pour vérifier si la valeur de *qauto* influence la stabilité du système, nous proposons d'analyser quatre résultats d'exécution du modèle, illustrées par la Figure 10.4 :

- chaque histogramme représente le nombre cumulé d'agents qui ont le même nombre de changement de niveaux, sur les 500 dernières itérations ;
- le nombre de changement est représenté par la valeur de *Fi2*, que nous analysons sur l'intervalle $[0, 20]$ - une valeur de *Fi2* proche de 0 signifie que l'agent effectue peu de

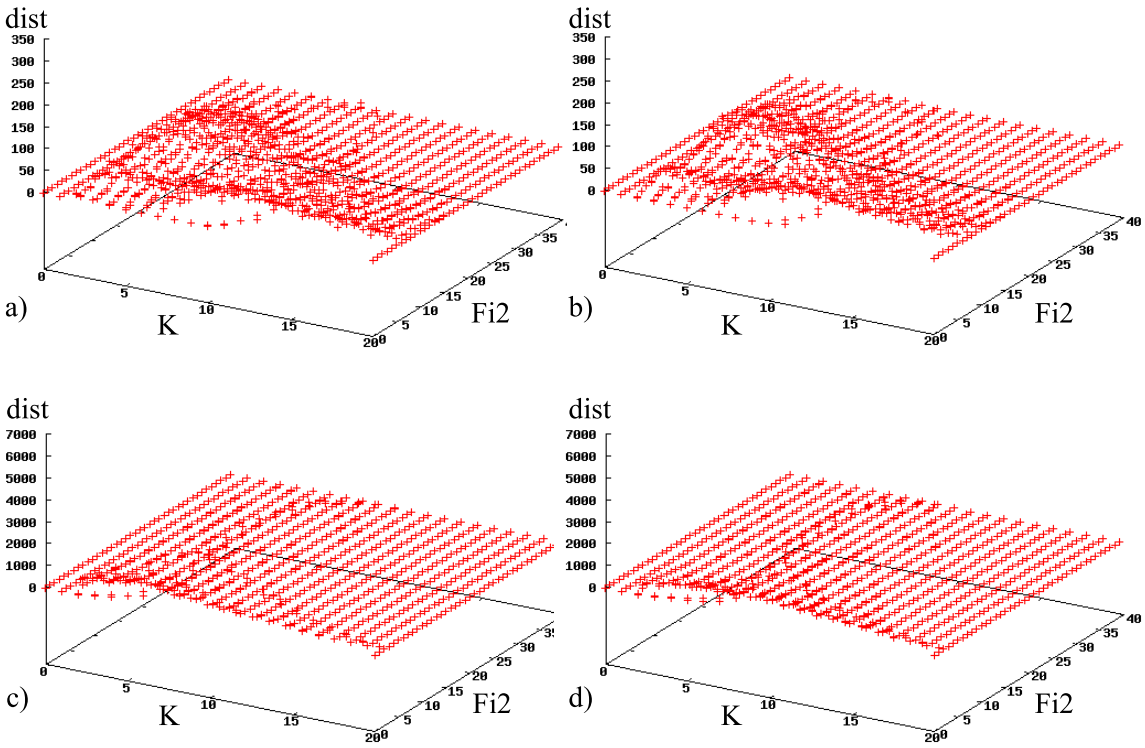


FIG. 10.4 – Résultats de simulation, analyse A.

changements, alors qu'une valeur proche de 20 représente une oscillation ;

Les histogrammes sont calculés sur l'ensemble des exécutions telles que :

- *qlevel*, *deltaPT*, et S varient sur l'ensemble des intervalles définis dans la description du cadre expérimental ;
- a) et c) sont calculés pour *qauto*= 1 (auto-stabilisation uniquement), alors que b) et d) sont calculés pour *qauto*= 0.5 (stabilisation mixte) ;
- a) et b) sont calculés pour 5 agents, c) et d) sont calculés pour 40 agents ;
- K est le paramètre analysé ;
- *Fi2* représente les mouvements d'un agent dans les 500 dernières itérations : une valeur proche de 0 signifie une absence d'oscillation ;
- *dist* représente le nombre cumulé d'agents qui vérifient le couple de valeurs défini par K et *Fi2*.

Observations Trois observations peuvent être notées, à partir de la Figure 10.4 :

- nous ne notons pas de différence significative entre les exécutions basées sur une auto-stabilisation uniquement et une stabilisation mixte ;
- les variations de *dist* atteignent des valeurs de *Fi2* supérieures à 20 avec 5 agents alors quelles restent inférieures à 15 avec 40 agents dans le système, ceci traduit une sensibilité plus importante aux variations de K pour les petites communautés d'agents ;
- l'intervalle des valeurs de K correspondant à des valeurs de *Fi2* proches de 0 semble différent selon la taille de la communauté d'agents.

Cette dernière observation nécessite une analyse plus détaillée de l'influence de K sur les oscillations du système.

Influence de K

Analyse B Afin de compléter les observations basées sur l'analyse A, nous proposons d'analyser le comportement du mécanisme de répartition en fonction du paramètre K. Rappelons que ce paramètre modélise la position du point d'inflexion dans la fonction de probabilités qui détermine si un agent doit changer de niveau de contrôle.

Les histogrammes de la Figure 10.5 reprennent les résultats d'analyse des exécutions du modèle du système selon les paramètres suivants :

- *deltaPT*, *qlevel* et S varient sur l'ensemble des intervalles définis dans la présentation du cadre expérimental ;
- *qauto*= 1, nous avons choisi de ne considérer que les configurations basées sur une auto-stabilisation du système ;
- lors de chaque exécution du modèle, le nombre d'agents dans le système est de 5 pour a), 15 pour b), et 40 pour c) ;
- K est le paramètre analysé ;
- *Fi2* représente les mouvements d'un agent dans les 500 dernières itérations : une valeur proche de 0 signifie une absence d'oscillation ;
- *dist* représente le nombre cumulé d'agents qui vérifient le couple de valeurs définies par K et *Fi2*.

Observations La première observation confirme l'influence du point d'inflexion K dans l'apparition d'oscillations dans la structure du système. Ainsi, pour une valeur de K autour de 15, la part d'agents « stables » croît significativement. Une autre observation peut être formulée. En effet, le début de la zone « stable » diffère suivant la taille de la population d'agents.

Valeur de K

Analyse C L'analyse B nous a permis d'identifier la relation entre l'apparition de phénomènes oscillatoires et la valeur du paramètre K. Afin d'évaluer l'attraction d'un niveau de la hiérarchie sur un agent en fonction de K, nous proposons dans l'analyse C, de mesurer le rapport entre les nombres de pas de simulation pendant lesquels un agent a occupé les deux niveaux les plus fréquentés. En conséquence, nous pouvons déduire qu'un niveau de la hiérarchie attire un agent, quand ce rapport (noté *ratio*) a une valeur « significativement » élevée.

La Figure 10.6 présente les résultats de l'analyse C pour les configurations suivantes du modèle :

- *qlevel* et S varient sur les intervalles définis dans la description du cadre expérimental,
- *qauto*= 1, nous ne considérons que les configurations du modèle basées sur une auto-stabilisation ;
- $deltaPT < 3$ implique que les agents des configurations a) et c) ont une puissance homogène, alors que $deltaPT > 4$ illustre des configurations où la puissance des agents est hétérogène dans b) et d) ;
- le système est composé de 5 agents pour les configurations représentées sur a) et b), et de 40 agents pour c) et d) ;
- K est le paramètre dont on souhaite évaluer l'influence ;
- *ratio* représente le rapport entre les durées passées dans les deux niveaux les plus fréquentés pour un agent donné ;
- *dist* est le nombre cumulé d'agents sur toutes les exécutions, qui vérifient le couple de valeurs définies par K et *ratio*.

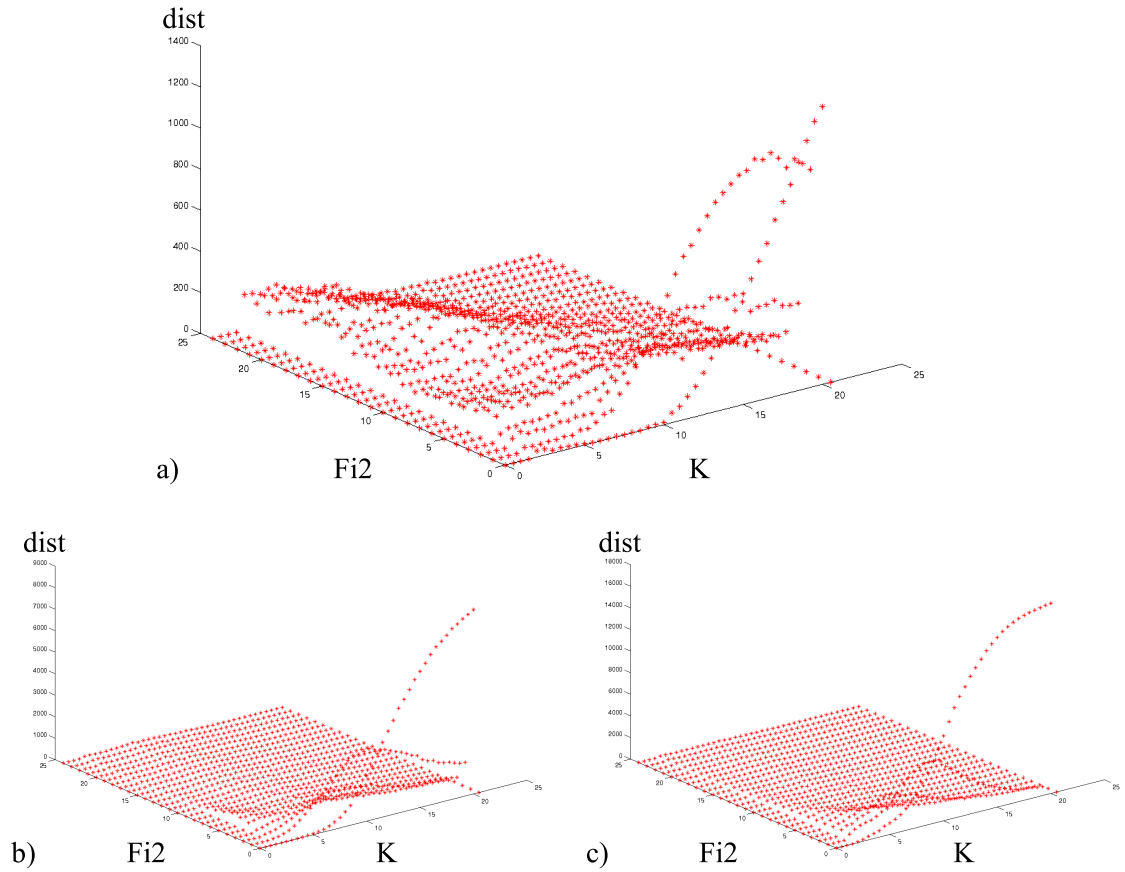


FIG. 10.5 – Résultats de simulation, analyse B.

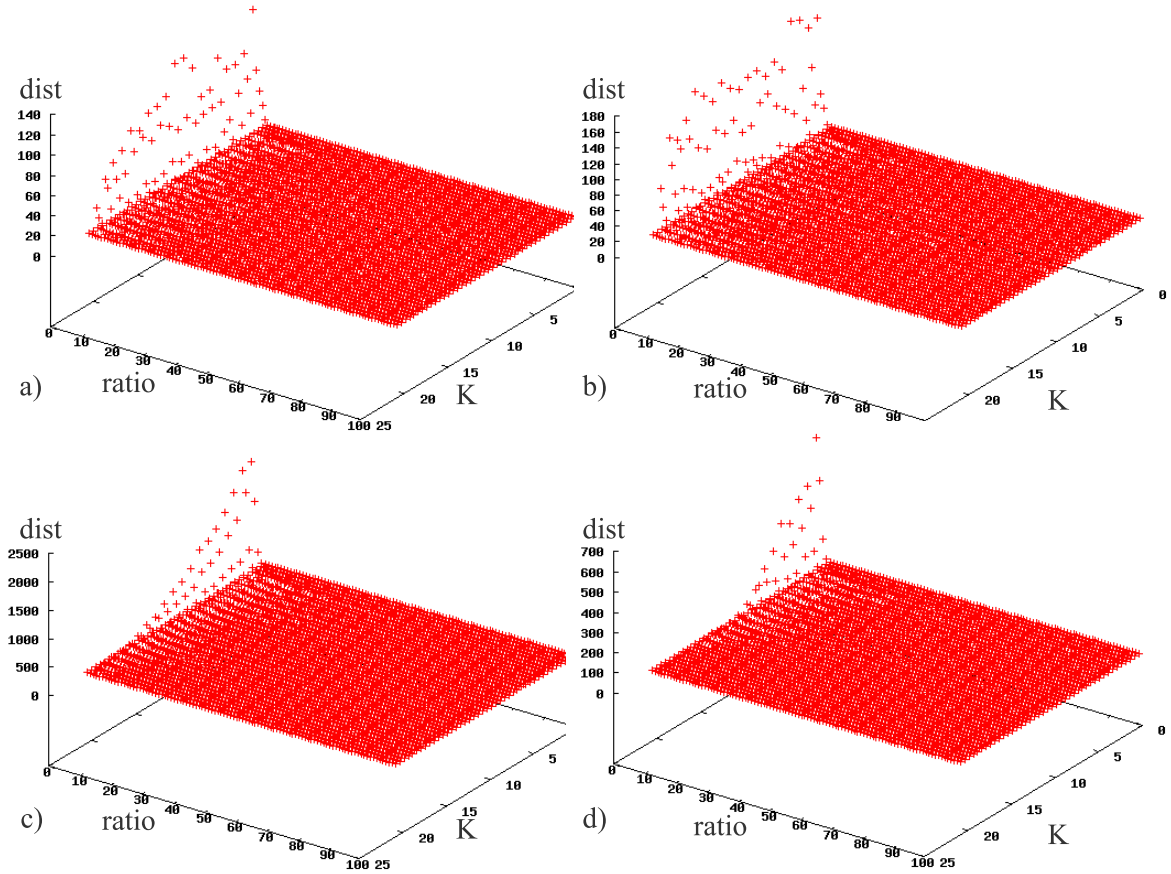


FIG. 10.6 – Résultats de simulation, analyse C.

Observations Le résultat de cette analyse laisse penser que le rapport entre les durées passées dans les deux niveaux les plus fréquentés est supérieur à 100 pour la majorité des valeurs possibles de K. Cette observation implique alors que l'attraction du niveau le plus fréquenté est très forte et nous supposons qu'un autre phénomène intervient. En effet, nous pensons que l'attraction observée dans cette analyse correspond à une stagnation du système. Dans un tel cas, le système n'évolue pas et ne sort pas de son état initial. En conséquence nous proposons la dernière analyse qui permet d'évaluer la vitalité de la structure de contrôle.

Vitalité du système

Analyse D Le dernier comportement que nous souhaitons analyser concerne la stagnation de la structure du système contrôle. Celle-ci apparaît lorsque le système ne sort plus d'un de ses états, notamment son état initial. Pour évaluer quelles sont les configurations qui engendre ce phénomène, l'analyse D propose de mesurer le rapport *ratio* déjà défini, et d'en déduire si au moins deux niveaux différents sont atteints par les agents du système. Pour cela, nous avons fixé un plafond au delà duquel une valeur de *ratio* implique une stagnation de l'agent.

La Figure 10.7 présente les résultats des exécutions des configurations telles que :

- S varie sur l'ensemble de l'intervalle défini dans la description du cadre expérimental ;
- $qauto = 1$, nous ne considérons que les configurations du modèle basée sur une auto-stabilisation ;

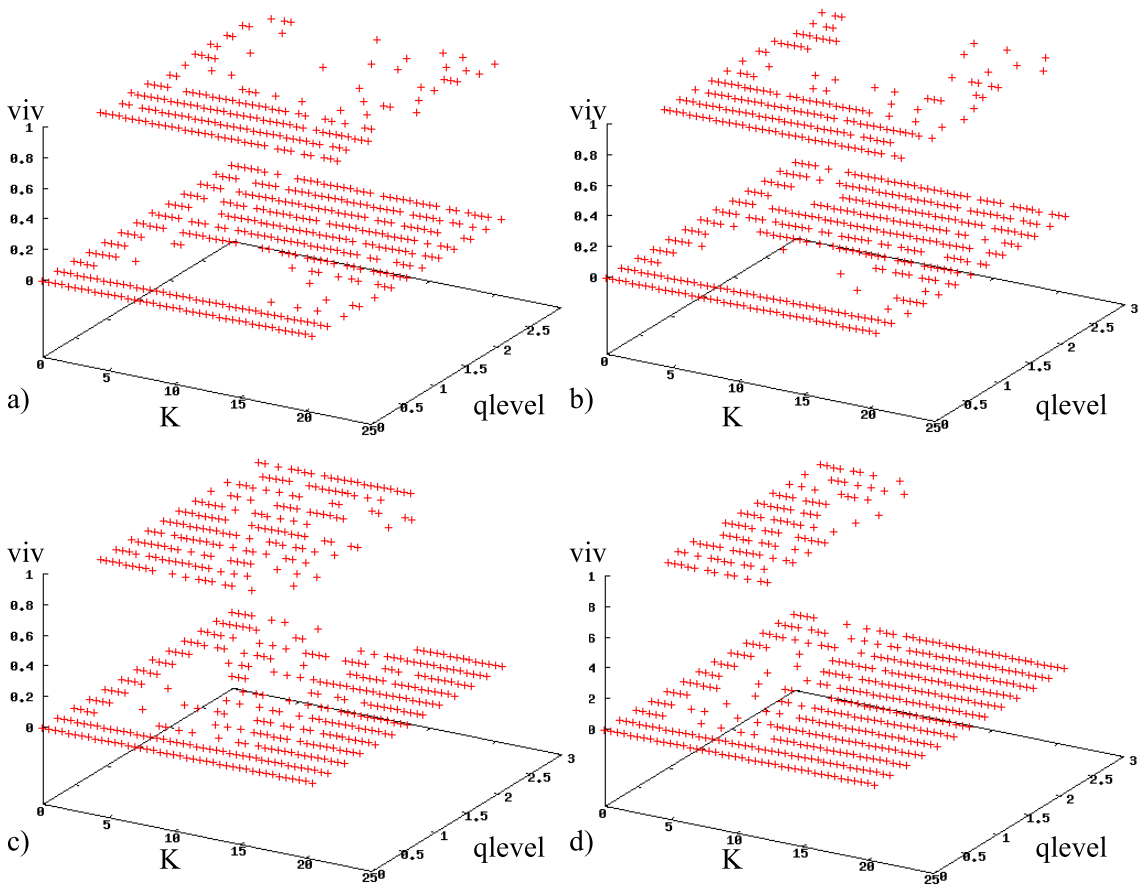


FIG. 10.7 – Résultats de simulation, analyse D.

- *deltaPT*< 3 pour a) et c), et *deltaPT*> 4 pour b) et d) ;
- le système est composé de 5 agents pour les configurations représentées sur a) et b), et de 40 agents pour c) et d) ;
- K et *qlevel* sont les deux paramètres dont on souhaite mesurer l'influence ;
- *viv* vaut 1 si le système évolue, et 0 s'il stagne.

Observations Dès les premiers coups d'oeil nous nous rendons compte qu'une part importante des configurations possibles n'évoluent pas et stagnent au niveau initial. En outre, cette analyse souligne aussi la différence de comportement rencontrée entre des systèmes dont la taille de la communauté d'agents et l'homogénéité sont significativement différentes. Dans les cas a) et c), les agents ont un rapport de puissances d'au plus 1/2, alors que dans les cas b) et d) le rapport peut aller jusqu'à 1/6. Enfin, la zone de valeurs de K et *qlevel*, correspondant à une structure de contrôle « vivante », présente des contours irréguliers ce qui laisse présager certaines difficultés lors du paramétrage du mécanisme sur une implémentation physique du système.

10.4 Synthèse

Les observations issues des analyses que nous avons menées lors de la campagne de simulation nous permettent d'identifier les trois résultats suivants.

Le premier résultat concerne la comparaison effectuée lors de l'analyse A. En effet, nous pouvons constater que, contrairement au modèle bio-inspiré initial, notre mécanisme d'organisation de la hiérarchie de contrôle n'oscille pas d'avantage s'il est basé sur une stabilisation mixte. Ainsi, des études complémentaires devront être effectuées sur le système physique que nous implémentons, afin de corréler les résultats issus des simulations et le comportement du système réel.

Nous avons également noté l'adaptation nécessaire de la valeur des paramètres K et *qlevel* en fonction de la nature de la configuration du système. Deux voies sont alors envisageables, car nous pouvons d'une part énoncer une relation entre la valeur de ces paramètres et le nombre et la puissance des agents dans le système, ou alors vérifier si les constantes introduites dans le modèle sont adaptées. C'est par exemple le cas de la constante K_w qui modélise l'intervalle de temps d'inhibition du mouvement d'un agent. La taille de cette intervalle pourrait par exemple être définie selon la puissance d'un agent. Cette piste devra être approfondie lors de l'implémentation du mécanisme sur le système physique.

Enfin, nous avons également noté qu'une importante part des configurations simulées présente un phénomène de stagnation de l'architecture de contrôle. Alors que les paramètres du mécanisme d'organisation dépendent de la configuration du système, il conviendra, lors de l'implémentation physique du système, d'ajouter des guides lors du paramétrage du mécanisme. En effet, un tel paramétrage sera nécessaire et nous pensons l'automatiser pour que le système s'organise sans notre supervision.

Troisième partie

Mise en œuvre

FIG. 10.8 – Logo de eZFusion, prononcé « easy fusion ».

Cette partie du mémoire est consacrée à la mise en œuvre de nos travaux. Elle présente eZFusion, prononcé « easy fusion », le logiciel utilisé comme cadre de développement et de test des algorithmes que nous proposons.

eZFusion est organisé autour de trois objectifs : la description, le développement et le déploiement d'un processus de fusion, afin de réaliser un contrôle sur le processus de fusion et sur son exécution.

Les architectures visées par notre logiciel vont du *smart phone* au PC *stand alone* en passant par les *box* Internet, dès lors que celles-ci sont reliées par un réseau.

La version courante d'eZFusion est un démonstrateur qui sert actuellement de cadre de déploiement à des processus de fusion répartis. Certains objectifs, pour l'heure hors de portée, seront atteints dans le prochain stade de développement du logiciel. En effet, un prototype d'eZFusion sera disponible fin 2009 pour une première phase de tests en laboratoire. Les projets déjà initiés seront consolidés et nous diffuserons notre logiciel au printemps 2010.

La suite de ce mémoire introduit eZFusion dans une présentation rapide. Les chapitres suivants détaillent l'actuelle implémentation.

Chapitre 11

Implémentation actuelle du système

Introduction

eZFusion implémente actuellement un partie des avancées théoriques de cette thèse. La première concerne la distinction entre le système de contrôle et le système reparti de fusion d'informations. Concernant les tâches de contrôle décrites dans la partie plus théorique du mémoire, eZFusion permet pour l'heure les points suivants :

- le déploiement d'une configuration du processus de fusion sur des ressources réparties,
- la reconfiguration du système pendant l'exécution du processus,
- la surveillance des performances du système, notamment par les mesures fournies par les sondes internes au système,
- et le contrôle du processus de fusion par l'adaptation de la valeur des paramètres des éléments de fusion, pendant l'exécution du processus.

Les développements actuels concernent la traduction automatique d'une configuration dans un modèle à réseau de Petri stochastique généralisé, dont des indices de performance sont évalués par le logiciel Smart [CJMS01].

Bien loin d'un catalogue de détails techniques, la suite de ce chapitre présente nos choix d'implémentation. En effet, lors du développement de eZFusion, plusieurs directions s'offraient à nous et nous discutons chaque alternative dans les paragraphes suivants.

11.1 Présentation générale

11.1.1 Compatibilité visée

Nos travaux et leurs résultats ont été menés pour un certain cadre applicatif. En l'occurrence, nous visons les systèmes composés d'éléments disponibles au grand public. Les architectures comme les nœuds de capteurs ou les capteurs intelligents, ainsi que les grappes de serveurs ou les grilles de calcul, sortent du cadre applicatif de nos travaux.

11.1.2 Systèmes d'exploitations compatibles

Outre le périmètre des systèmes visés, l'implémentation de nos résultats suppose la disponibilité d'une machine virtuelle Java et d'un canal de communication TCP entre les nœuds du système.

La machine virtuelle Java est nécessaire à l'environnement d'exécution de eZFusion. En effet, notre logiciel est développé dans le contexte de plates-formes OSGi [OSG] communicantes. À

l'heure actuelle, eZFusion a été testé sous l'implémentation Knopflerfish 4.0.4 d'OSGi [Kno] et les communications sont basées sur les services présentés dans le Paragraphe 11.5.

11.1.3 Infrastructure réseau

Au niveau de la communication entre les plates-formes OSGi, les problèmes de sécurité, de confidentialité et d'intégrité des données sortent du cadre de nos recherches. Par ailleurs, les problématiques de routage des communications sortent également du périmètre de notre étude. L'actuelle version d'eZFusion base les communications sur deux types de protocoles. Il convient donc de s'assurer que les protocoles HTTP et RMI sont disponibles.

11.1.4 Installation et utilisation

eZFusion est un logiciel publié sous licence GNU-GPL[1], gratuit d'installation et d'utilisation.

Où trouver eZFusion ? Deux dépôts sont mis à la disposition des utilisateurs qui souhaitent installer eZFusion :

- dépôt tout public : `http://ezfusion.sourceforge.net/` ;
- dépôt recherche : `http://www.polytech.univ-savoie.fr/listic/`

Le dépôt tout public détaille ce chapitre du mémoire et plusieurs exemples d'utilisation d'eZFusion sont disponibles au téléchargement. Ce dépôt est maintenu à jour par les auteurs et développeurs d'eZFusion. Le second dépôt est quand à lui maintenu à jour par le laboratoire à l'origine d'eZFusion. Ce dépôt orienté recherche regroupe les publications qui exploitent eZFusion.

11.2 Cadre de déploiement de eZFusion

La première question à laquelle nous avons été confrontée concerne le choix d'un cadre de déploiement de notre système. Dès le début de notre réflexion, nous avons identifié les points suivants :

- le développement de notre système doit être indépendant d'une utilisation de eZFusion sous Windows, Linux et Mac OS,
- le cadre de déploiement doit permettre l'adaptation du système pendant son exécution,
- le développement doit tirer partie d'éléments déjà disponibles.

Dès lors que ces contraintes sont identifiées, le choix d'un développement sous Java paraît raisonnable. De plus, nous souhaitons mettre l'accent sur l'aspect dynamique du système et non sur la performance maximale du système, où le bénéfice d'un langage interprété est alors moins évident.

Parmi les cadres de déploiement disponibles en Java, nous avons identifié deux types de solutions : les serveurs d'applications et les plates-formes OSGi [OSG]. Ces deux types de solutions permettent le déploiement de systèmes logiciels sans redémarrage du serveur et facilitent également l'adaptation des « services » déployés. Il nous est alors apparu que la gestion automatique d'un service est plus aisée sous OSGi. En effet, nous souhaitons qu'une fois installé, le système soit le plus autonome possible. Dès lors, comme nous le verrons dans la suite, la création des unités de déploiement et la gestion des services peuvent être gérées depuis l'intérieur de la plate-forme.

En outre, rappelons que nos travaux ont été initiés en septembre 2006, et les plates-formes OSGi étaient de plus en plus présentes dans l'environnement de développement des systèmes dynamiques. Notre intuition que OSGi allait devenir incontournable dans les années à venir se

[1] http ://www.gnu.org/licenses/gpl-3.0.txt

vérifie aujourd'hui. En effet, il est fréquent de voir une *box* Internet héberger un système Linux équipé d'une plate-forme OSGi, ou un compteur électrique déployer une interface de gestion de l'énergie d'un foyer, également sous OSGi.

Enfin, OSGi est aussi une solution à présent éprouvée et de nombreux services sont disponibles et facilement réutilisables. Ce dernier avantage a finalement orienté notre choix vers ce cadre de déploiement.

Concernant le choix de l'implémentation que nous avons retenue, aucune ne se distinguait significativement des autres au début de nos travaux et nous avons préféré garder l'implémentation Knopflerfish.

11.3 Conditionnement du déploiement

Une plate-forme OSGi définit plusieurs niveaux de granularité pour un système déployé. En effet, même si l'entité de déploiement est le *bundle*, un logiciel peut être déployé sur plusieurs *bundles* et dépendre d'un nombre totalement différent de services. Cette correspondance entre eZFusion, des *bundles* et des services est une ambiguïté que nous avons dû lever lors du développement de notre système.

11.3.1 *Bundles*

Comme l'illustre la Figure 11.1, le conditionnement de eZFusion en terme de *bundle* a évolué depuis le début de nos travaux. Il est passé de trois à un *bundle* pour les raisons suivantes.

3 *bundles* Le premier conditionnement envisagé découpait notre système en trois *bundles* dédiés respectivement au système d'exécution, au système de contrôle et aux éléments communs dans le système (comme les interfaces des services pour ne citer qu'un exemple). Le but de cette organisation était de représenter physiquement l'indépendance des deux sous-systèmes, le troisième *bundle* n'étant alors qu'une conséquence de ce choix. Ainsi, lors de nos premiers tests, nous avons effectivement réalisé l'intérêt de pouvoir modifier un des sous-système sans impacter l'autre. Pourtant, nous n'avons pas retenu ce conditionnement, principalement car la procédure de mise à jour et de redéploiement des trois *bundles* était fastidieuse lors de la correction d'erreur de développement.

1 *bundle* Ce conditionnement est celui actuellement adopté. Nous avons choisi de regrouper tous les éléments de eZFusion au sein d'un seul *bundle* pendant la phase de développement. L'application des corrections du code du système est ainsi facilitée, notamment lors de la mise à jour de l'ensemble de notre plate-forme de tests. En effet, le système que nous proposons est par nature réparti, et les scénarios que nous déployons lors de nos tests nécessitent l'utilisation d'un parc informatique de trois à dix cadres d'exécution. L'inconvénient majeur du regroupement des sous-systèmes est qu'il rend impossible tout redéploiement partiel. Ce type de redéploiement de *bundle* nous permettrait pourtant de gérer des défaillances d'une partie du système.

2 *bundles* C'est parce que le regroupement en un seul *bundle* ne permet pas l'indépendance des sous-systèmes de contrôle et d'exécution, que nous comptons, à l'issue de la phase de développement, conditionner eZFusion dans deux *bundles*. Nous pensons qu'avec une telle organisation, les sous-systèmes serons, d'une part, indépendants au sein d'une plate-forme OSGi, mais nous pensons également qu'un *bundle* « système de contrôle » pourrait d'autre part gérer plusieurs *bundles* « système d'exécution » déployés sur autant de plates-formes.

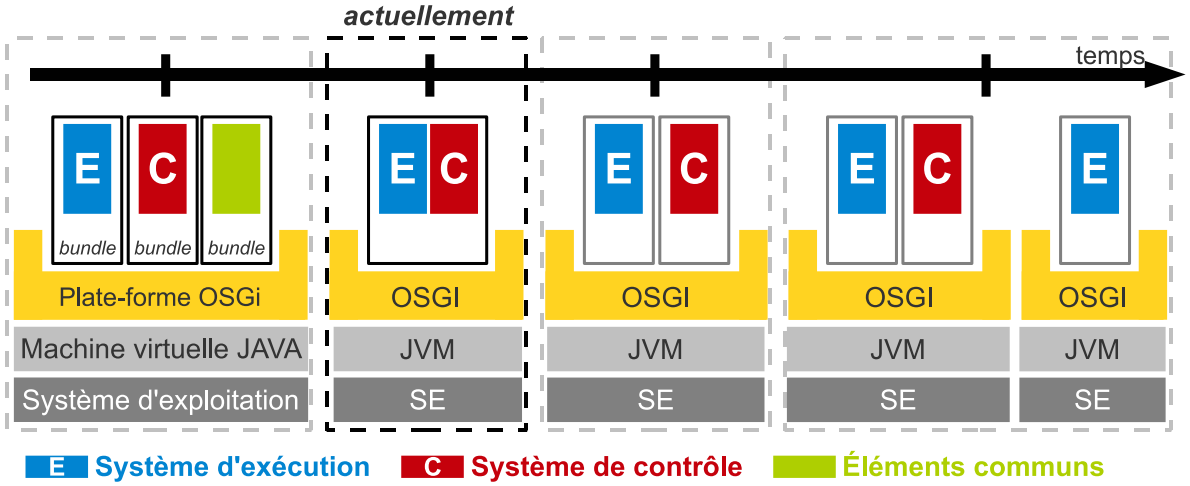


FIG. 11.1 – Conditionnements de eZFusion dans un cadre d'installation OSGi.

Au regard de l'évolution des services inclus dans les plates-formes OSGi, il est raisonnable de penser que ce type d'organisation sera transparent du point de vue des sous-systèmes. En effet, l'intégration et le partage de services entre plates-formes OSGi est de plus en plus facilité. Ainsi, la création d'une plate-forme OSGi « virtuelle », déployée sur plusieurs PC, n'est plus du domaine de la spéculation et devrait faire son apparition dans les années à venir.

11.3.2 Services

Alors que le nombre de découpages possibles d'eZFusion en terme de *bundle* reste assez limité, le nombre de services que le système peut déployer au sein d'une plate-forme OSGi n'a de limite que celle du développeur.

Ainsi, nous avons choisi de limiter le déploiement de services et de ne proposer, à terme, qu'un unique service pour le système de contrôle. En effet, la philosophie de eZFusion sera alors de déployer plusieurs systèmes d'exécution qui dépendront alors de système de contrôle déjà déployés. Avec une telle structure, les cadres d'exécution qui ne déploient qu'un système d'exécution s'enregistreront et proposeront leurs ressources au reste du système.

Les autres services actuellement déployés par eZFusion sont autant de dépendances à enregistrer avec des services déjà proposés par la communauté. Ainsi, les services de messagerie ou de *log* n'ont pas pour but de rester dans le système.

11.4 Éléments d'une configuration

La description que nous faisons d'une configuration du système, dans le Chapitre 6, laisse assez de liberté pour introduire plusieurs types d'organisations structurelles dans l'implémentation du système d'exécution. Ainsi, comme l'illustre la Figure 11.3, nous avons été confrontés au questionnement suivant :

- où déployer les nœuds de fusion ?
- est-il utile d'utiliser des *wires* pour relier les nœuds de fusion ?

Le mot *wire* désigne un objet manipulé dans le cadre du service *wire admin*[OSG09]. Nous reviendrons sur ce point à la fin de ce paragraphe.

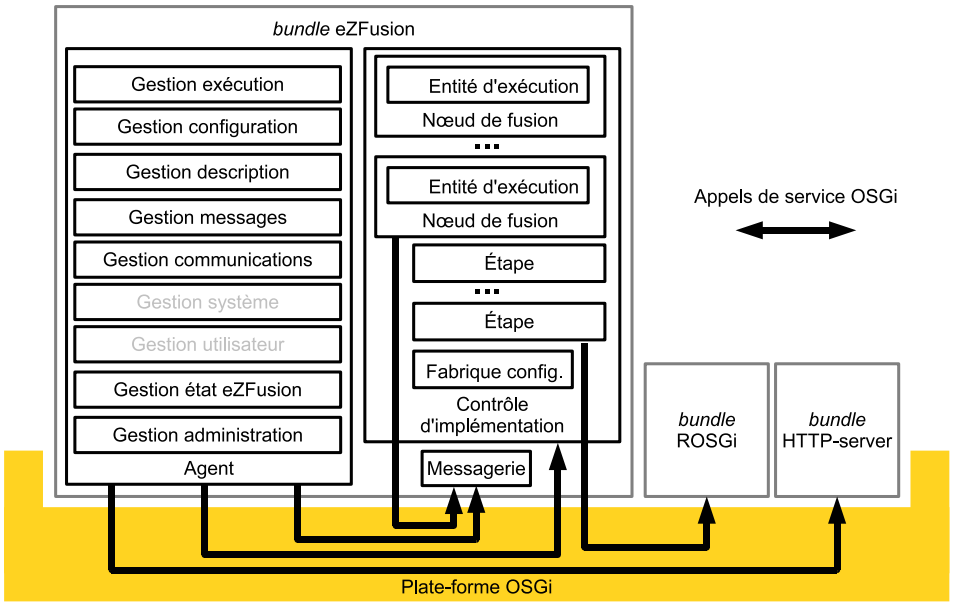


FIG. 11.2 – Composants et services de eZFusion dans un cadre d'installation OSGi.

11.4.1 Déploiement des nœuds de fusion

Nœuds de fusion internes Cette organisation est celle actuellement utilisée pour déployer les nœuds de fusion d'une configuration. Ainsi, les objets qui implémentent les éléments d'une configuration sont stockés dans le *bundle* du système d'exécution.

Cette organisation nous a mis face à des problèmes de chargement de classes. En effet, les nœuds de fusion sont des objets génériques, indépendants des classes utilisées pour implémenter les fonction de fusion. En outre, celles-ci sont *a priori* inconnues et indisponibles au moment de l'installation de eZFusion au sein de la plate-forme, il est donc impossible de spécifier dans les caractéristiques de déploiement d'eZFusion, quels sont les éléments Java requis lors du fonctionnement : le fichier `manifest` du *bundle* ne peut donc pas être complété.

Pour contourner cette limitation, nous avons choisi d'utiliser un chargeur de classe interne à eZFusion, ce qui nous permet d'instancier les nœuds de fusion avec des éléments externes à la plate-forme OSGi alors que les *bundles* de notre système sont déjà lancés.

Afin d'intégrer eZFusion au mieux dans les plates-formes OSGi, nous souhaitons déployer les nœuds de fusion déployés dans le système dans des *bundles* dédiés pour deux raisons : éviter d'utiliser un mécanisme de chargement de classe maison et « publier » les nœuds de fusion sous la forme de services de fusion contrôlés par notre système, mais aussi utilisés par d'autres systèmes.

Nœuds de fusion indépendants Cette organisation des nœuds de fusion est diamétralement opposée à celle actuelle retenue. En effet, alors que tous les nœuds de fusion sont déployés dans le même *bundle* dans la première solution, ici chaque nœud est déployé au sein d'un *bundle* qui lui est associé. Les éléments de contrôle de eZFusion ne sont plus en charge du chargement des classes qui implémentent les fonctions de fusion. Nous avons déjà testé ce type d'organisation, où notre système crée automatiquement un nouveau *bundle* dans le système de fichier puis le déploie dans la plate-forme OSGi. Ce *bundle* peut alors inclure tous les éléments nécessaires à l'instanciation d'un nœud de fusion, notamment par des informations plus précises dans le fichier `manifest` du *bundle*.

Alors qu'il est déjà possible d'enregistrer des services de fusion dans la première organisation,

celle-ci identifie la correspondance entre un *bundle* et un service.

L'inconvénient majeur de cette solution réside dans la duplication des éléments Java nécessaires au déploiement des nœuds de fusion. En effet, lors de nos tests, nous avons noté que ces éléments sont le plus souvent communs à plusieurs nœuds de fusion. En conséquence, nous proposons une dernière organisation possible.

Familles de nœuds de fusion Cette dernière organisation des nœuds de fusion découle de l'expérience acquise pendant le développement de notre système. Ainsi, nous avons exhibé les limitations d'un regroupement des nœuds de fusion dans un seul *bundle*. Néanmoins, une organisation opposée, où chaque nœud prend place dans un *bundle*, présage un problème de redondance dans le système d'exécution. De plus, cette redondance des éléments Java, communs à des ensembles de nœuds de fusion et *a priori* indisponibles, n'est pas toujours nécessaire.

En effet, nous pensons que notre système doit permettre, dans sa prochaine version, la définition de familles de déploiement composées :

- des codes exécutables qui implémentent les fonctions de fusion de la famille,
- des archives Java complémentaires requises lors de l'exécution des codes du point précédent,
- des nœuds de fusion vides, qui accueillent les codes exécutables du premier point.

De là, chaque famille prend place dans un *bundle* automatiquement construit lors du déploiement d'une configuration.

Une autre organisation reste possible. En effet, un nœud de fusion pourrait être indépendant et déployé dans un seul *bundle*, et les éléments Java communs à une famille de nœuds seraient regroupés dans un *bundle* spécial. La résolution des classes disponibles ne serait donc plus du ressort du *bundle* mais de la plate-forme OSGi.

11.4.2 Déploiement d'une étape locale

Rappel Une étape locale correspond à l'implémentation d'un lien telle que les éléments de fusion e et e' connectés par ce lien sont affectés au même cadre d'exécution. Tous ces termes sont précisés dans le Chapitre 6 de ce mémoire.

Wire admin Le mot *wire* désigne ici un objet qui connecte un « producteur » à un « consommateur » : tous ces termes sont définis dans le cadre du service *wire admin* [OSG]. L'intérêt d'un *wire* réside dans sa connexion automatique aux producteurs ou aux consommateurs déployés dans la plate-forme, chaque extrémité de ce lien définit alors un type d'élément échangé - ce type est également repris lors de la définition des producteurs et des consommateurs. Ainsi, si un couple (p_1, c_1) est relié par un *wire*, la déconnexion d'une des extrémités va induire une recherche des autres producteurs ou consommateurs compatibles au sein de la plate-forme.

Si initialement nous avons implémenté les étapes locales par des *wires*, nous avons écarté ce choix dès lors que nous maîtrisions la création ou la destruction des nœuds de fusion. Il n'était donc pas nécessaire de disposer d'un comportement automatique du lien entre les nœuds de fusion. Ce choix n'est pour autant pas définitif et l'adoption d'un déploiement externe des nœuds de fusion peut justifier un retour à l'utilisation de *wires*.

11.5 Communication

Dans la version actuelle de notre système, les communications sont de trois natures. Les échanges de résultats partiels sont synchronisés, ce qui n'est pas le cas des messages de contrôle transmis au sein d'un cadre d'exécution ou entre cadres d'exécution. Tous ces points sont détaillés dans la suite de ce paragraphe.

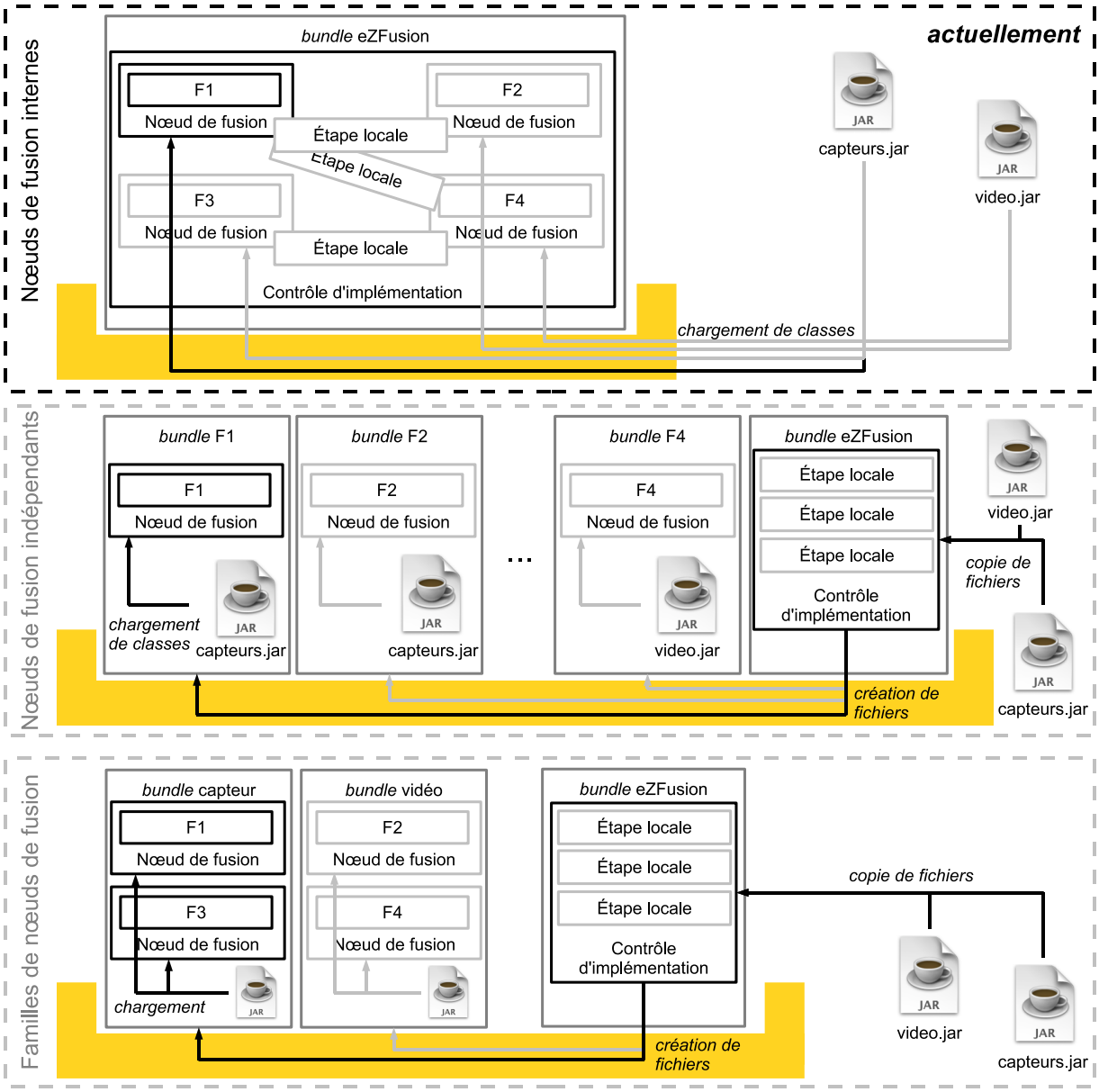


FIG. 11.3 – Organisation du déploiement des nœuds de fusion dans eZFusion : dans cet exemple F1 et F3 sont des sources d'informations utilisées par F2 et F4.

11.5.1 Exécution du processus

Intra-cadre d'exécution Le premier type de communication ne fait pas intervenir d'élément externe au cadre d'exécution. En effet, lorsque le transfert d'un résultat partiel doit être réalisé entre deux nœuds de fusion d'un même cadre, la référence en mémoire de l'objet qui représente ce résultat est échangée *via* un appel de méthode. De plus, l'élément qui relie deux nœuds de fusion implémente un lien entre deux ports de communication. Comme un nœud de fusion peut définir plusieurs ports de communication, plusieurs lien peuvent être connectés à nœud de fusion donné. Ainsi, deux nœuds de fusion fixés peuvent être reliés par un ensemble d'étapes locales, définies pour chaque couple de ports connectés.

Inter-cadre d'exécution Le second type de communication permet l'échange d'un résultat partiel entre deux nœuds de fusion déployés sur deux cadres d'exécution. La mise en œuvre de ce transfert n'avait de limite que notre imagination : de l'ouverture de *socket* jusqu'à la publication d'un service Web. La solution que nous avons retenue, *Remote OSGi* [RAR07], propose de créer un relais (i.e. un *proxy*) sur une plate-forme B, d'un service déployé sur une plate-forme A. L'utilisation du service relais est ainsi transparente sur B. La mise en œuvre du canal de communication peut en outre être paramétrée selon les limitations des cadres d'exécution : par exemple l'impossibilité d'utiliser un appel RMI.

Plusieurs éléments de notre système étaient alors candidat à ce partage entre deux cadres d'exécution. La première solution testée fût celle où deux nœuds de fusion ($F1$ le producteur et $F2$ le consommateur) étaient déployés sur deux cadres et un relais de $F2$ était déployé sur le même cadre que $F1$. Cette solution, *a priori* satisfaisante, fût vite abandonnée à cause d'effets imprévus. Ainsi, pour chaque lien entre des ports de $F1$ et $F2$, un relais était installé sur le cadre de $F1$. De plus, si $F2$ consommait des résultats produits par un autre nœud de fusion également sur un autre cadre, le système déployait autant de copies de $F2$ que de liens implémentés.

Nous avons donc sélectionné un autre élément du système pour un déploiement de type R-OSGi. Le seul élément possible était l'étape locale qui relie deux nœuds de fusion. Ainsi, comme nous l'illustrons dans la Figure 11.4, une étape qui relie deux nœuds initialement sur le même cadre peut être en partie déconnectée. L'extrémité déconnectée est alors exploitée sur le cadre distant où un relais de cette étape a été créé.

L'indépendance entre le nombre de relais déployés pour un même élément et le nombre de nœud de fusion est l'autre intérêt majeur de cette solution. En effet, quelque soit le nombre de ports et le nombre de « producteurs » connectés à un nœud donné, un unique relais est installé pour chaque étape locale partiellement connectée.

11.5.2 Contrôle du système

Les communications de contrôle prennent deux formes dans notre système : les ordres et les messages informatifs. Les ordres sont émis par le système de contrôle en direction du système d'exécution et les messages informatifs le contraire.

De plus, dans notre modèle de déploiement nous posons que chaque tâche de contrôle est répartie sur l'ensemble du système. Les communications à l'intérieur d'une tâche et entre les tâches de contrôle doivent donc être implémentées par des échanges de messages sur le réseau. Au contraire, les transferts de messages informatifs peuvent être limités au cadre d'exécution.

Deux types de média de contrôle sont utilisés suite à ce constat. Le premier média est réalisé par un service d'échange de messages au sein de la plate-forme OSGi. Ce service est basé sur le principe d'abonnement à un sujet sur lequel des objets peuvent écrire et réceptionner des messages. Par exemple, nous utilisons ce service pour la mise à jour des paramètres des nœuds de fusion, transmis directement depuis le système de contrôle. À l'inverse, l'état d'un nœud de

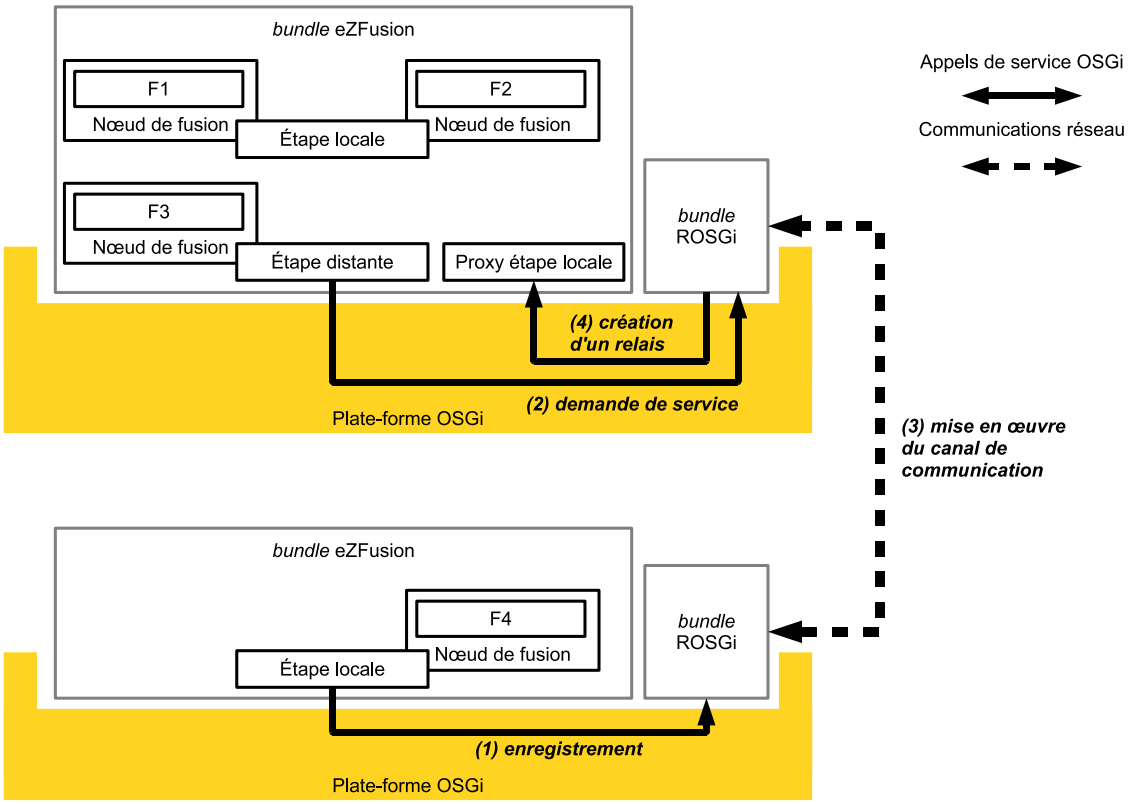


FIG. 11.4 – eZFusion dans un cadre d'installation OSGi : mise en œuvre de deux liens, de F1 vers F2 en local et de F3 vers F4 grâce à ROSGi.

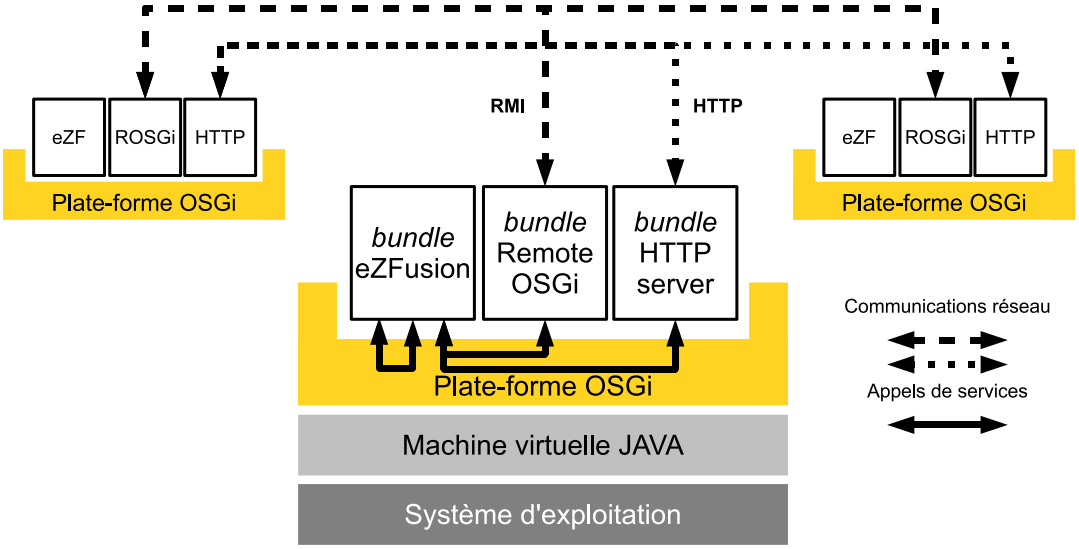


FIG. 11.5 – eZFusion dans un cadre d'installation OSGi : protocoles de communication.

fusion est posté puis lu par le système de contrôle. Enfin, les tâches du système de contrôle échangerons des messages *via* ce service lorsqu'elles sont activées par le même cadre d'exécution.

Les communications entre cadres d'exécution ont suivi plusieurs évolutions lors du développement de notre système. Si l'utilisation de R-OSGi pour créer un relais du service d'échange de messages semblait tentant, les problèmes d'identifications et de redondance des services entre les plates-formes demandaient des efforts d'adaptation conséquents. Durant cette phase de développement, nous avions également mis en place une interface de commande destinée à l'administration du système. Cette interface de commande, toujours en vigueur, repose sur des pages Web ouvertes par un navigateur. De là, l'idée fut d'utiliser ces interfaces pour l'échange de messages de contrôle entre les cadres. Ainsi, la majorité des messages reçus *via* HTTP sur un cadre d'exécution, ne sont plus émis par l'administrateur mais par d'autres cadres d'exécution.

11.6 Synthèse

eZFusion pose les bases de nos apports théoriques en contrôle des systèmes répartis de fusion d'informations. Notre logiciel reprend le modèle que nous avons défini dans les premiers chapitres de ce mémoire, et implémente certaines des tâches de contrôle.

L'état de développement actuel permet à eZFusion de déployer une configuration, c'est-à-dire l'affectation du processus de fusion, sur les ressources du système. De plus, il d'ores et déjà possible de reconfigurer le système pendant l'exécution du processus.

L'organisation de notre logiciel va évoluer pour tendre vers un système simple et autonome. S'il est simple de développer un élément de fusion dans le cadre de eZFusion, le contrôle du système, et notamment la description d'une configuration du système nécessite de la pratique. A terme, cette configuration sera automatiquement déduite des ressources détectées. Concernant la partie du système de contrôle qui intervient à la suite du déploiement d'une configuration, l'évaluation des performances du système est pour l'instant notre priorité.

D'un point de vue technique, nous constatons que le choix d'une installation de eZFusion au sein d'un réseau de plates-formes OSGi était judicieux. De nouveaux périphériques compatibles apparaissent tous les jours. De plus la puissance de calcul disponible sur chaque cadre d'exécution évolue à la hausse, allégeant ainsi l'impact d'une implémentation Java de notre outil.

Chapitre 12

Conclusion

Ce dernier chapitre de la thèse met en perspective nos travaux, nous présentons dans un premier temps un résumé de nos contributions (Paragraphe 12.1). L'intérêt de chaque contribution y est alors discuté au regard des travaux présentés dans les précédents chapitres.

Les modèles présentés dans cette thèse laissent encore des zones d'ombres délimitées dans le Paragraphe 12.2. Plusieurs éléments de notre système de contrôle restent à développer et nos travaux prospectifs sont autant de pistes à explorer dans les années à venir. La Figure 12.1 donne une représentation de l'état d'avancement de nos travaux. Les tâches de contrôle de notre système y sont représentées par des barres de progression : une barre au minimum signifie que la problématique est identifiée dans le contexte des systèmes répartis de fusion d'informations, alors qu'une barre complète représente l'implémentation de nos solutions. Il est alors facile de cerner les points qu'il nous reste à préciser ou à implémenter.

En outre, nous présentons en conclusion de ce mémoire, deux exemples de domaines d'ouverture dans lesquels notre système peut être réutilisé.

12.1 Contributions

À l'issue des trois années de cette thèse, nos travaux sur le contrôle des Systèmes Répartis de Fusion d'Informations (SRFI) ont abouti à la présentation de quatre modèles. Ces modèles s'inscrivent dans un système global dont l'implémentation, nommée eZFusion, est en cours de réalisation.

L'organisation de notre étude du contrôle des SRFI suit plusieurs étapes qui sont autant d'objectifs intermédiaires que nous avons atteints. Premièrement, il est naturel de penser qu'un panorama des systèmes et des modèles de fusion d'informations était la première pierre à sceller pour bâtir nos travaux. Dès lors que cette étude des travaux existants était effectuée, les contours de notre approche ont commencé à se dessiner.

À ce moment de notre étude, nous avons choisi de construire un modèle nous permettant de décrire un processus de fusion d'informations indépendamment du modèle théorique sous-jacent et en s'affranchissant des contraintes physiques liées au matériel.

Modèle de description et de déploiement d'un SRFI

Ce premier modèle permet à un expert en fusion d'informations, de décrire un processus composé de traitements élémentaires. Ces traitements, que nous définissons comme des éléments de fusion, décrivent des ports par lesquels transitent des informations au sein d'un graphe de flots de données.

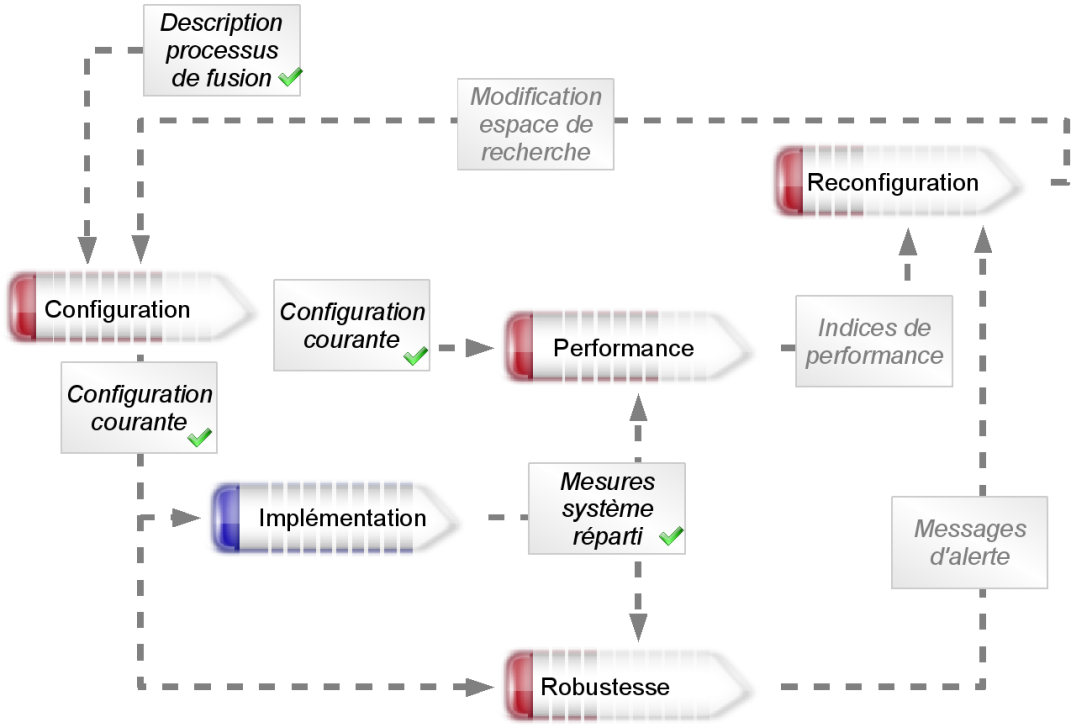


FIG. 12.1 – État d'avancement détaille du système de contrôle.

Le premier axe de contrôle que nous avons identifié à ce niveau, concerne la qualité du traitement appliqué aux informations. Notre modèle de description du processus de fusion a alors été étendu par l'ajout de ports dédiés au paramétrage du traitement. La valeur des paramètres ainsi décrits, est alors adaptée selon les recommandations du concepteur du processus de fusion. Une telle adaptation peut être effectuée manuellement, par la visualisation du résultat final puis par la correction de la valeur d'un paramètre, ou automatiquement, grâce à la description explicite de ces étapes par une boucle de contrôle.

Le déploiement d'un processus de fusion fait intervenir le calcul d'une configuration du système qui applique effectivement le traitement. Cette configuration est d'une part composée du processus de fusion, dont elle tire les éléments à déployer, et d'autre part des ressources physiques utilisées pour appliquer le traitement. Ces ressources prennent la forme de puissance de calcul, de canaux de communication, mais aussi d'archives contenant du code exécutable fourni par un développeur.

L'originalité de notre approche réside dans ce dernier point avec la séparation des rôles :

- d'un côté le concepteur du processus, expert en fusion d'informations, décrit un processus indépendamment du modèle de fusion sous-jacent,
- de l'autre côté le développeur traduit les traitements élémentaires en code exécutable, indépendamment des mécanismes de communication et de contrôle.

Cette dichotomie du travail de conception est, selon nous, une caractéristique indispensable aux futurs SRFI.

Dès lors qu'une configuration du système est définie, sa mise en œuvre est réalisée par une partie du système. En effet, lors de notre étude des SRFI existants, nous avons souvent constaté l'ambiguïté latente de la définition de la partie du système en charge du contrôle. Ainsi, lors de nos réflexions sur le déploiement et le contrôle d'un processus de fusion, nous avons choisi d'expliciter deux sous-systèmes, respectivement chargés de l'exécution du processus et du contrôle

du système.

Modèle du système de contrôle

Nous avons ainsi identifié quelle était la place du contrôle dans un SRFI. Alors qu'un premier axe de contrôle permet la gestion de la qualité du résultat produit par le système, un second axe concerne la gestion de la performance et de la robustesse du système. Ces deux points permettent au système de contrôle d'évaluer le bénéfice d'une reconfiguration du système afin d'en améliorer, par exemple, le temps de traitement, ou de gérer les ajouts et les suppressions de ressources pendant l'exécution du processus.

À l'issue de cette identification de ces deux buts du système de contrôle, les tâches de contrôle suivantes ont été mises en évidence :

- la tâche *Configuration* permet l'évaluation des configurations possibles du système pour un processus de fusion donné ;
- la tâche *Implémentation* met effectivement en œuvre un processus de fusion et fournit des mesures produites par des sondes internes au système ;
- la tâche *Analyse* évalue des indices de performance d'une configuration donnée ;
- la tâche *Robustesse* surveille la configuration courante du système ;
- la tâche *Reconfiguration* décide d'un changement de configuration du système.

Chaque nouvelle tâche identifiée s'accompagne d'un ensemble de problématiques, et nos solutions s'inspirent de celles issues de domaines différents du notre, quand elles existent.

Modèle d'analyse de performance

C'est par exemple le cas de notre modèle d'analyse de performance des SRFI. En effet, le but de ce modèle réside dans sa capacité à évaluer la performance d'une configuration *a priori*.

La structure du modèle du système est alors traduite en un réseau de Petri stochastique généralisé, à partir d'une configuration du système. Les valeurs des paramètres de ce modèle sont mesurées sur des sondes internes au système.

L'intérêt de notre approche repose sur la réutilisation de la vue du système physique, dérivée des sondes internes, avec plusieurs configurations possibles du processus de fusion. Ainsi, pour un même ensemble de ressources, nous effectuons plusieurs évaluations de configurations possibles que nous comparons pour ne sélectionner que celle effectivement déployée.

Modèle de répartition du système de contrôle

Alors que le système d'exécution est par nature réparti, nous proposons que le système de contrôle exploite également l'ensemble des ressources disponibles. Notre répartition du système de contrôle est ainsi fondée sur un système multi-agents au sein duquel deux communautés d'agents interagissent :

- les agents *mobiles* sont hébergés sur des architectures peu puissantes, comme par exemple un téléphone portable, et coopèrent avec la seconde communauté d'agents ;
- les agents *immobiles* sont par exemple hébergés sur des ordinateurs et disposent de ressources importantes.

La définition de ces deux communautés d'agents provient des deux types d'architectures (i.e. puissantes ou limitées) souvent rencontrées. Nous exploitons alors cette distinction des agents lors du déploiement du système où un agent mobile n'est exploité que pour la mise en œuvre du processus de fusion, alors que les agents immobiles sont privilégiés pour le contrôle du système dans une hiérarchie multi-niveaux.

De plus, nous proposons un mécanisme bio-inspiré d'auto-organisation de la hiérarchie du système de contrôle. Ce choix nous permet de gérer les fluctuations des caractéristiques du système dont ni le nombre d'agents, ni la nature des ressources, ne sont *a priori* disponibles. De surcroît, ces deux caractéristiques peuvent évoluer dans le temps.

eZFusion

eZFusion est le système que nous proposons pour déployer et contrôler un SRFI. Cet outil logiciel est destiné aux concepteurs de processus de fusion souhaitant mettre en œuvre ses travaux sur un ensemble dynamique de ressources. eZFusion est le fruit de travaux initiés dans [PBHM08] et poursuivis dans [WTS+08]. En effet, pendant cette thèse nous avons souhaité rester proches des solutions techniques disponibles, notamment en choisissant d'implémenter nos modèles dans le cadre de plates-formes OSGi. Ainsi, dans un souci d'évolution du système, eZFusion exploite des services déployés au sein de plates-formes OSGi. C'est par exemple le cas de R-OSGi, que nous utilisons pour mettre en œuvre les transferts de résultats issus du processus de fusion.

En outre, lors des différentes évolutions du système, nous avons différencie les éléments liés à l'implémentation de nos modèles de contrôle, de ceux liés à l'exploitation du cadre d'installation de eZFusion. Ces derniers éléments, propres au cadre d'installation de notre système, seront détaillés dans la documentation technique de eZFusion afin d'en faciliter la diffusion. Une autre implémentation de nos modèles pourrait par exemple être réalisée dans un cadre J2EE.

12.2 Perspectives

Les modèles que nous présentons dans ce mémoire n'apportent qu'une partie des solutions aux problèmes liées au contrôle des SRFI. Ainsi, nous proposons des pistes de recherche pour la gestion de la robustesse du système et pour le déclenchement d'une reconfiguration. D'autres travaux peuvent également approfondir nos modèles.

Extension du modèle d'analyse

Le modèle d'analyse de performance peut ainsi être réutilisé dans le cadre plus général des systèmes répartis. En outre, alors que nous exploitons une résolution centralisée du calcul des indices de performance du système, la répartition de cette analyse est une piste de travail. En effet, la structure du modèle du système, traduite d'une configuration que nous maîtrisons, peut impliquer des simplifications de l'espace de résolution des états du système.

Généralisation du processus déployé

Lors de l'élaboration de notre modèle de description, nous avons été amenés à formuler des restrictions sur la structure du processus que nous souhaitons contrôler. Le relâchement de certaines hypothèses, principalement sur le caractère acyclique du processus et sur la sémantique d'exécution, permettrait la conception et le contrôle d'autres types d'applications réparties.

En effet, nous pensons qu'une nouvelle famille d'applications réparties va se démocratiser dans les années à venir. Ainsi, alors que les architectures mobiles ne cessent de se miniaturiser, et que les besoins en puissance de calcul augmentent inexorablement, l'exploitation de ressources *a priori* inconnues peut ouvrir la voie à de nouveaux comportements. Voici un exemple qui illustre notre idée.

Cet exemple concerne un utilisateur équipé de l'équivalent d'un téléphone portable, installé dans ses habits et ses bijoux. La puissance de calcul embarquée est limitée, pour autant cet utilisateur peut dialoguer en direct sur Internet. Ce scénario propose l'utilisation de l'équipement de

l'utilisateur comme de périphériques intégrés dans le système réparti déployé dans le commerce ou le transport emprunté. Un tel type d'applications permettrait d'une part la mutualisation des ressources de calcul et d'autre part limiterait la puissance d'émission des dispositifs de l'utilisateur : le point de connexion au système le plus proche serait alors suffisant.

Contrôle du système de contrôle

Enfin, il n'aura pas échappé au lecteur que notre système de contrôle peut être vu comme un processus de fusion. Ainsi, comme nous l'avons évoqué lors de l'introduction du Chapitre 5, un niveau de contrôle supérieur peut être défini. Un des objectifs de ce contrôle de haut niveau serait par exemple l'élaboration du processus de fusion en fonction des éléments de traitement disponibles.

Si l'on reprend l'exemple précédent, nous pouvons imaginer que l'utilisateur, lors de sa première visite d'un lieu, demande au système un « guide du voyageur » à partir des informations disponibles sur le réseau. Dès lors que le but du processus est défini, le système de contrôle procède à l'élaboration du processus de fusion d'informations adéquat. Notre système entre alors en jeu et permet le déploiement de ce processus, quelques soient les ressources accessibles à l'utilisateur.

Bibliographie

[ALRL04] Avizienis A., Laprie J.-C., Randell B., and Landwehr C. Basic concepts and taxonomy of dependable and secure computing. *IEEE Transactions on Dependable and Secure Computing*, 1 :11–33, 2004.

[APPJ08] Abras Shadi, Ploix Stephane, Pesty Sylvie, and Jacomino Mireille. *A Multi-agent home automation system for power management*, volume 15, chapter 1, pages 59–68. Springer Berlin Heidelberg, 30 May 2008.

[BBH07] Borne Pierre, Benrejeb Mohamed, and Haggège Joseph. *Les réseaux de neurones : présentation et applications*, volume 15. Méthodes et techniques de l'ingénieur, ophrys edition, 2007.

[BCD+05] Bhatti Shah, Carlson James, Dai Hui, Deng Jing, Rose Jeff, Sheth Anmol, Shucker Brian, Gruenwald Charles, Torgerson Adam, and Han Richard. MANTIS OS : An Embedded Multithreaded Operating System for Wireless Micro Sensor Platforms. *Mobile Networks and Applications*, 10(4) :563–579, 24 June 2005.

[BD95] Baeijs Christof and Demazeau Yves. Les organistations dans les systèmes multi-agents. In *Proceedings of 4ème Journée Nationales du PRC-IA sur les Systèmes Multi-Agents*, Toulouse, December 1995.

[BDPP06] Bouyssou Denis, Dubois Didier, Pirlot Marc, and Pomerol Jean-Charles. *Concepts et méthodes pour l'aide à la décision : outils de modélisation*, volume 1. Hermes science, 2006.

[Bez81] Bezdek James C. *Pattern Recognition with Fuzzy Function Algorithms*. Plenum Press, 1981.

[BHK+06] Balan Rahul, Han Chih-Chieh, Kumar Rengaswamy Ram, Tsigkogiannis Ilias, and Srivastava Mani. Multi-level software reconfiguration for sensor networks. In *Proceedings of ACM Conference on Embedded Systems Software (EMSOFT)*, pages 112–121, Seoul, Korea, October 2006.

[BHMP07] Benoit Eric, Huget Marc-Philippe, Moreaux Patrice, and Passalacqua Olivier. Integrating OPC data into GSN Infrastructure. Technical report, LISTIC - University of Savoie, 1 October 2007.

[BHMP09] Benoit Eric, Huget Marc-Philippe, Moreaux Patrice, and Passalacqua Olivier. Reconfiguration of distributed Information Fusion System - A case study. In *Proceedings of DCDS Dependable Control of Discrete Systems*, pages 309–314, Bari, Iatly, June 2009.

[Blo05] Bloch Isabelle. Fusion d'informations numériques : panorama méthodologique. In *Proceedings of Journées Nationales de la Recherche en Robotique 2005*, pages 78–88, October 2005.

[BP02] Blash Erik P. and Plano Susan. JDL level 5 fusion model "user refinement" issues and applications in group tracking. In *Proceedings of SPIE, Signal processing, sensor fusion, and target recognition XI*, volume 270-279, July 2002.

[BQX08] Biswas Pratik K., Qi Hairong, and Xu Yingyue. Mobile-agent-based collaborative sensor fusion. *Information fusion*, 9(3) :399–411, July 2008.

[Bub05] Bubnicki Zdzislaw. *Modern control theory*. Springer, 2005.

[CCL07] Chao-Lin Wu, Chun-Feng Liao, and Li-Chen Fu. Service-Oriented Smart-Home Architecture Based on OSGi and Mobile-Agent Technology. *Systems, Man, and Cybernetics, Part C : Applications and Reviews, IEEE Transactions on*, 37(2) :193–205, March 2007.

[CES04] Culler David, Estrin Deborah, and Srivastava Mani. Overview of Sensor Networks. *Computer*, 37(8) :41–49, 2004.

[CGM+92] Carpenter G. A., Grossberg S., Markuzon N., Reynolds J. H., and Rosen D. B. Fuzzy ARTMAP : a neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE transcations on neural networks*, 3(5) :698–713, 1992.

[CJMS01] Ciardo G., Jones R. L., Miner A. S., and Siminiceanu R. SMART : Stochastic Model Analyzer for Reliability and Timing. In *Tools of Aachen 2001 International Multi-conferenc on Measurement, Modeling and Evaluation of Computer Communication Systems*, pages 29–34, Aachen, Germany, September 2001.

[Edi09] Edition Larousse. Dictionnaire en ligne. www.larousse.fr, 2009.

[Fer97] Ferber Jacques. *Les systèmes multi-agents - vers une intelligence collective*. InterEditions - Masson édition, August 1997.

[Flo99] Flores-Mendez Roberto A. Towards a standardization of multi-agent system framework. *Crossroads*, 5(4) :18–24, 1999.

[Fou02] Foundation for intelligent physical agents. FIPA Contract Net Interaction Protocol Specification. Technical report, FIPA, 3 December 2002.

[GB92] Guan Jiwen W. and Bell David A. *Evidence theory and its applications*. North-Holland, 1992.

[Goo09] Google map service. Google map. maps.google.com, 2009.

[Ham06] Hamilton Peter F. *L'étoile de Pandore*. Bragelonne, 2006.

[HKS+03] Han Chih-Chieh, Kumar Ram, Shea Roy, Kohler Eddie, and Srivastava Mani. A Dynamic Operating System for Sensor Nodes. In *Proceedings of MobiCom'03 : 9th annual international conference on Mobile computing and networking*, pages 81–95, 2003.

[HKS04] Hitha Alex, Kumar Mohan, and Shirazi Behrooz. MidFusion : An adaptive middleware for information fusion in sensor network applications. *Intelligent Sensors, Sensor Networks and Information Processing Conference*, pages 617, 622, 2004.

[HMM+07] Horré Wouter, Matthys Nelson, Michiels Sam, Joosen Wouter, and Verbaeten Pierre. A survey of middleware for wireless sensor networks. Technical report, Department of Computer Science, K.U.Leuven, Katholieke Universiteit Leuven, B-3001 Heverlee (Belgium), August 2007.

[IM04] Ilyas Mohammad and Mahgoud Imad. *Handbook of sensor networks : compact wireless and wired sensing systems*. CRC Press, 2004.

[JVN05] Janakiram D., Venkateswarlu R., and Nitin S. COMiS : Component Oriented Middleware for Sensor Networks. In *Proceedings of 14th IEEE Workshop on Local Area and Metropolitan Networks (LANMAN)*, September 2005.

[Kle92] Kleer Johan. Focusing on Probable Diagnosis. *Readings in modelbased Diagnosis*, 1992.

[Kno] Knopflerfish. Knopflerfish OSGi. http ://www.knopflerfish.org/.

[KTW99] Kokar Mieczyslaw, Tomasik Jerzy, and Weyman Jerzy. A formal approach to information fusion. In *Proceedings of 2nd Intern. Conf. On Information Fusion*, volume 1, 1999.

[KTW04] Kokar Mieczyslaw, Tomasik Jerzy, and Weyman Jerzy. Formalizing classes of information fusion systems. *Information Fusion*, 5(3) :189–202, September 2004.

[KZK97] Kam Moshe, Zhu Xiaoxun, and Kalata Paul. Sensor Fusion for Mobile Robot Navigation. In *Proceedings - IEEE*, volume 85, pages 108–119. IEEE Institute of electrical and electronics, 1997.

[Lap95] Laprie Jean-Claude (Collectif). *Le guide de la sûreté de fonctionnement.* Editions Cépaduès, 1995.

[LBMV08] Le-Normand Nicols, Boissiere Julien, Meger Nicolas, and Valet Lionel. Supply chain management by ùeans of FLM-rules. In *Proceedings of 12th European Conf. on Principles and Practice of Knowledge Discovery in Databases PKDD'08*, pages 29–36, September 2008.

[LC02] Levis Philip and Culler David. Maté : a tiny virtual machine for sensor networks. In *Proceedings of ASPLOS-X : Proceedings of the 10th international conference on Architectural support for programming languages and operating systems*, pages 85–95, 2002.

[LCK+97] Liggins II Martin E., Chong Chee-Yee, Kadar Ivan, Alford Mark G., Vannicola Vincent, and Thomopoulos Stelios. Proceedings of distributed Fusion Architectures and Algorithms for Target Tracking. In *proceedings of IEEE*, January 1997.

[LGMK05] Laforge Bertrand, Guez David, Martinez Michael, and Kupiec Jean-Jacques. Modeling embryogenesis and cancer : an approach based on an equilibrium between the autostabilization of stochastic gene expression and the interdependence of cells for proliferation. *Progress in Biophysics and Molecular Biology*, 89 :93–120, September 2005.

[Lli07] Llinas James. New challenges for defining information fusion requirements. In Springer, editor, *Proceedings of Information fusion and geographic information systems : proceedings of the Third International Workshop, Lecture notes in geoinformation and cartography*, pages 1–16, 2007.

[LMG+04] Levis Philip, Madden Samuel, Gay David, Polastre Joseph, Szewczyk Robert, Woo Alec, Brewer Eric A., and Culler David E. The emergence of networking abstractions and techniques in tinyos. In *Proceedings of First symposium on networked systems design and implementation (NSDI'04)*, pages 1–14, March 2004.

[LW81] Lehrer K. and Wagner C. *Rational Consensus in Science and Society.* Dordrech, 1981.

[M.01] Diaz M., editor. *Les réseaux de Petri.* Traité IC2. Hermes, Paris, 2001.

[MBD98] Marsan M. A., Bobbio A., and Donatelli S. Petri nets in performance analysis : an introduction. Reisig and Rozenberg, 1998. 211-256.

[MBW+04] Makarenko A., Brooks A., Williams S., Durrant-Whyte H., and Grocholsky B. A decentralized architecture for active sensor networks. *Robotics and Automation*, 2 :1097–1102, April 2004.

[McC81] McConway K. J. Marginalization and linear opinion pools. *Journal of the American Statistical Association*, 76 :410–414, 1981.

[MD05] Marin Cristina and Desertot Mikael. Sensor bean : a component platform for sensor-based services. In ACM International Conference Proceeding Series, editor, *Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*, volume 115, 2005.

[MD06] Makarenko Alexei and Durrant-Whyte Hugh. Decentralized Bayesian algorithms for active sensor networks. *Information Fusion*, 7(4) :418–433, December 2006.

[Men42] Menger Karl. Statistical Metrics. *National Academy of Sciences*, 28(12) :535–537, December 1942.

[MGG84] Ajmone Marsan M., Balbo G., and Conte G. A class of generalized of stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on computer systems*, 2(2) :93–122, May 1984.

[Mit98] Mitchell Melanie. *An introduction to genetic algorithms, Complex adaptive systems.* MIT Press, 1998.

[MR04] Meger Nicolas and Rigotti Christophe. Constraint-based mining of episode rules and optimal window sizes. In *Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 313–324, 2004.

[New01] New Digital Group. Smarty template engine. http ://www.smarty.net, 2001.

[OSG] OSGi Alliance. http ://www.osgi.org.

[OSG09] OSGi Alliance. Wire Admin Service. OSGi Service Plateform Release Version 4.2 Compendium Specification. www.osgi.org/Specifications/, September 2009. Pages 193-235.

[PBHM08] Passalacqua Olivier, Benoit Eric, Huget Marc-Philippe, and Moreaux Patrice. Integrating OPC data into GSN infrastructure. In *Proceedings of IADIS International Conference Applied Computing 2008*, April 2008.

[PC03] Parsons Olga and Carpenter Gail A. ARTMAP neural networks for information fusion and data mining : map production and target recognition methodologies. *Neural networks*, 16 :1075–1089, 2003.

[PON05] Pavlin, Oude, and Nunnink. A MAS approach to fusion of heterogeneous information. *Web Intelligence, 2005. Proceedings. The 2005 IEEE/WIC/ACM International Conference on*, pages 802–804, September 2005.

[PWN09] Pernestal Anna, Warnquist Hakan, and Nyberg Mattias. Modeling and troubleshooting with interventions applied to an auxiliary truck braking system. In *Proceedings of DCDS09 Dependable Control of Discrete Systems*, pages 285–290, 10 June 2009.

[RAR07] Rellermeyer Jan S., Alonso Gustavo, and Roscoe Tomothy. R-OSGi : Distributed Applications Through Software Modularization. In *Lecture Notes in Computer Science*, volume 4834/2007, pages 1–20. Springer Berlin / Heidelberg, 1 November 2007.

[RNL03] Ruiz L. B., Nogueira J. M., and Loureiro A. A. F. MANNA : a management architecture for wireless sensor networks. *Communications Magazine IEEE*, 41(2) :116–125, February 2003.

[Sal07] Salehi Ali. *The GSN book.* 14 April 2007. http ://gsn.svn.sourceforge.net/.

[Sas02] Sasiadek J. Z. Sensor fusion. *Annual Reviews in Control*, 26(2) :203–228, 2002.

[SBW98] Steinberg A. N., Bowman C. L., and White F. E. Revision to the JDL data fusion model. In *Proceedings of Joint NATO/IRIS Conference*, October 1998.

[Sha76] Shafer Glenn. *A Mathematical Theory of Evidence.* Princeton University Press, 1976.

[Shi99] Shimanuki Y. OLE for process control (OPC) for new industrial automation systems. In *Proceedings of 1999 IEEE International Conference onSystems, Man, and Cybernetics. IEEE SMC'99 Conference Proceedings.*, volume 6, pages 1048–1050, October 1999.

[Sme90] Smets P. The combination of Evidence in the Transferable Belief Model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5) :447–458, 1990.

[Smi80] Smith Reid G. The Contract Net Protocol : High-Level communication and control in a distributed problem solver. *IEEE Transtactions on computers*, C-29(12) :1104–1113, December 1980.

[SPU99] Solaiman B., Pierce L. E., and Ulaby F. T. Multisensor data fusion using fuzzy concepts : application toland-cover classification using ERS-1/JERS-1 SAR composites. *Geoscience and Remote Sensing, IEEE Transactions on*, 37(Part 1) :1316–1326, May 1999.

[SS05] Schweizer B. and Sklar A. *Probabilistic Metric Spaces.* Dover Publications, 2005.

[Sun06] Sun Shu-Li. Multi-sensor optimal fusion fixed-interval Kalman smoothers. *Information Fusion*, August 2006.

[Wal02] Wald Lucien. *Data Fusion : definitions and architectures - fusion of images of different spatial resolutions.* Ecole des mines de Paris, 2002.

[Whi96] White J. Mobile agent white paper. General Magic Inc., 1996.

[Wil96] Williamson James R. Gaussian ARTMAP : a neural network for fast incremental learning of noisy multidimensional maps. *Neural Networks*, 9(5) :881–897, 1996.

[WTS+08] Wan Khairunizam, Todo Atsushi, Sawada Hideyuki, Passalacqua Olivier, Benoit Eric, Huget Marc-Philippe, and Moreaux Patrice. Video conference smart room : an information fusion system based on distributed sensors. In *Proceedings of Mecatronics2008*, May 2008.

[Yao05] Yao Yiyu. Web Intelligence : New Frontiers of Exploration. In *Proceedings of the 2005 International Conference on Active Media Technology*, pages 3–8, Takamatsu, Kagawa, Japan, 19 May 2005.

[YRBL06] Yu Yang, Rittle Loren J., Bhandari Vartika, and LeBrun Jason B. Supporting concurrent applications in wireless sensor networks. In ACM Press, editor, *Proceedings of SensSys'06 : the 4th international conference on Embedded networked sensors systems*, pages 139–152, 2006.

[ZAR05] Zhu Qiuming, Aldridge Stuart L., and Resha Tomas N. Hierarchical Collective Agent Network (HCAN) for efficient fusion and management of multiple networked sensors. *Information Fusion*, 8(3) :266–280, July 2005.

[ZKY96] Zadeh Lotfi Asker, Klir George J., and Yuan Bo. *Fuzzy sets, fuzzy logic, and fuzzy systems.* World Scientific, 1996.

[ZPK00] Zeigler Bernard P., Praehofer Herbert, and Kim Tag Gon. *Theory od modeling and simulation. Integrating discrete event and continuous complex systems.* Academic Press, second edition edition, 2000.

Quatrième partie

Annexes

Chapitre 13

Premiers pas avec eZFusion

Introduction

Cette partie du manuscrit est consacrée à la mise en œuvre de nos travaux. Elle présente eZFusion, prononcé « easy fusion », le logiciel utilisé comme cadre de développement et de test pour les algorithmes que nous proposons.

EZFusion est organisé autour de trois objectifs : la description, le développement et le déploiement d'un processus de fusion, afin de réaliser un contrôle sur le processus de fusion et sur son exécution.

Les architectures visées par notre logiciel vont du *smart phone* au PC *stand alone* en passant par les *box* Internet, dès lors que celles-ci sont reliées par un réseau.

La version courante d'eZFusion est un démonstrateur qui sert actuellement de cadre de déploiement à des processus de fusion répartis. Certains objectifs, pour l'heure hors de portée, seront atteints dans le prochain stade de développement du logiciel. En effet, un prototype d'eZFusion sera disponible fin 2009 pour une première phase de tests en laboratoire. Les projets déjà initiés seront consolidés et nous diffuserons notre logiciel au printemps 2010.

La suite de ce manuscrit introduit eZFusion dans une présentation rapide. Les chapitres suivants détaillent l'actuelle implémentation du système et un cas d'utilisation clôt cette partie du manuscrit.

13.1 Compatibilité visée

Nos travaux et leurs résultats ont été menés pour un certain cadre applicatif. En l'occurrence, nous visons les systèmes composés d'éléments disponibles au grand public comme un PDA, un *smart-phone*, une *box* Internet, un *media-center*, un PC portable ou de bureau et un serveur PC ou en lame. Les architectures plus petites, comme les nœuds de capteurs ou les capteurs intelligents, ainsi que les unités de calcul plus larges, comme les grappes de serveurs ou les grilles

FIG. 13.1 – Logo de eZFusion, prononcé « easy fusion ».

de calcul, sortent du cadre applicatif de nos travaux.

13.1.1 Systèmes d'exploitations compatibles

Outre le périmètre des systèmes visés, l'implémentation de nos résultats suppose la disponibilité d'une machine virtuelle Java et d'un canal de communication TCP entre les nœuds du système.

La machine virtuelle Java est nécessaire au cadre d'installation de eZFusion. En effet, comme l'illustre la Figure 11.5, notre logiciel est développé dans le contexte de plates-formes OSGi [OSG] communicantes. À l'heure actuelle, eZFusion a été testé sous l'implémentation Knopflerfish 4.0.4 d'OSGi [Kno] et les communications sont basées sur les services présentés dans le Paragraphe 11.5.

13.1.2 Infrastructure réseau

Au niveau de la communication entre les plates-formes OSGi, les problèmes de sécurité, de confidentialité et d'intégrité des données sortent du cadre de nos recherches. Par ailleurs, les problématiques de routage des communications, au niveau du graphe de connexion des cadres d'exécution, sortent également du périmètre de notre étude.

L'actuelle version d'eZFusion base les communications sur deux types de protocoles. Il convient donc de s'assurer que les protocoles HTTP et RMI sont disponibles.

13.2 Installation et utilisation

eZFusion est un logiciel publié sous licence GNU-GPL[1], gratuit d'installation et d'utilisation.

Où trouver eZFusion ? Deux dépôts sont mis à la disposition des utilisateurs qui souhaitent installer eZFusion :

- dépôt tout public : `http://ezfusion.sourceforge.net/`
- dépôt recherche : `http://www.polytech.univ-savoie.fr/listic/`

Le dépôt tout public fournit plusieurs exemples d'utilisation d'eZFusion sont disponibles au téléchargement. Ce dépôt est maintenu à jour par les auteurs et développeurs d'eZFusion. Le second dépôt est quand à lui maintenu à jour par le laboratoire à l'origine d'eZFusion. Ce dépôt orienté recherche regroupe les publications qui utilisent eZFusion.

Comment installer eZFusion ? eZFusion doit être installé comme un *bundle* OSGi. Pour cela, il suffit de télécharger l'archive Java disponible sur les dépôts du projet et de l'installer sur la plate-forme OSGi. Il est également possible de l'installer directement par référence à partir de l'URL de téléchargement comme un *bundle* en ligne.

Si le service ROSGi n'est pas déjà installé, veuillez télécharger le fichier `remote_rosgi.jar` disponible sur les dépôts du projets et l'installer *avant* eZFusion.

Pour installer un bundle sur une implémentation Knopflerfish d'OSGi, veuillez suivre les étapes suivantes :

1. lancer la plate-forme OSGi : `java -jar framework.jar`
2. cliquer sur *File* puis *Open Bundle File...*
3. sélectionner le fichier `mon_bundle.jar` puis valider par *Open*
4. démarrer le *bundle* en le sélectionnant dans la partie gauche de l'interface puis en cliquant sur *Bundles* et *Start*.

[1] http ://www.gnu.org/licenses/gpl-3.0.txt

```
Exported services
#36 org.ezfusion.logger.WebLoggerServlet
#40 org.ezfusion.execution.Controller
    eZfusion_all_in_one
#44 javax.servlet.http.HttpServlet
#41 org.ezfusion.agent.servlet.SysStatusServlet
#47 org.ezfusion.msgsrv.MsgClient
#37 org.ezfusion.logger.Logger
    eZfusion_all_in_one
#46 javax.servlet.http.HttpServlet
#43 org.ezfusion.agent.UserGUIInt
#38 org.ezfusion.msgsrv.MsgServer
    eZfusion_all_in_one
#42 org.ezfusion.agent.manager.ROSGiTest
    R_OSGi Remote Service
#45 javax.servlet.http.HttpServlet
#39 org.ezfusion.msgsrv.MsgClient
Imported services
#40 org.ezfusion.execution.Controller
    eZfusion_all_in_one
#37 org.ezfusion.logger.Logger
    eZfusion_all_in_one
#38 org.ezfusion.msgsrv.MsgServer
    eZfusion_all_in_one
#18 org.osgi.service.http.HttpService
    eZfusion_all_in_one
```

FIG. 13.2 – Liste des services OSGi importés et exposés par eZFusion.

13.3 Premiers pas

Ce paragraphe présente les actions de base de la version démonstrateur de eZFusion. Même si le fonctionnement général du logiciel ne devrait pas évoluer, les futures corrections et améliorations du système peuvent avoir un impact sur les paragraphes suivants. Il convient alors de se référer aux documentations publiées sur les différents dépôts du projet.

13.3.1 Démarrage de eZFusion

Après l'installation de notre logiciel au sein d'une plate-forme OSGi, le démarrage d'eZFusion s'effectue par l'activation du *bundle* (commande `start` suivie du numéro du *bundle*). Si toutes les dépendances de services sont résolues, c.f. la Figure 13.2, le *bundle* est démarré et l'interface Web d'eZFusion est accessible à l'adresse suivante : `http://127.0.0.1:8080/ezfusion`

L'interface de eZFusion (Figure 13.3) est découpée en deux parties, respectivement pour les commandes et pour les informations.

Interface de commande Le menu déroulant en haut de l'interface de commande permet de sélectionner l'adresse de l'interface réseau utilisée par le système de contrôle. En effet, plusieurs interfaces peuvent être activées sur une cadre d'exécution et le système de contrôle peut ne répondre qu'à certaines d'entre elles. Les cinq menus disponibles dans l'interface de commande sont :

– *Tests* : ce menu est utilisé pour tester l'activation du service ROSGi sur un cadre distant,

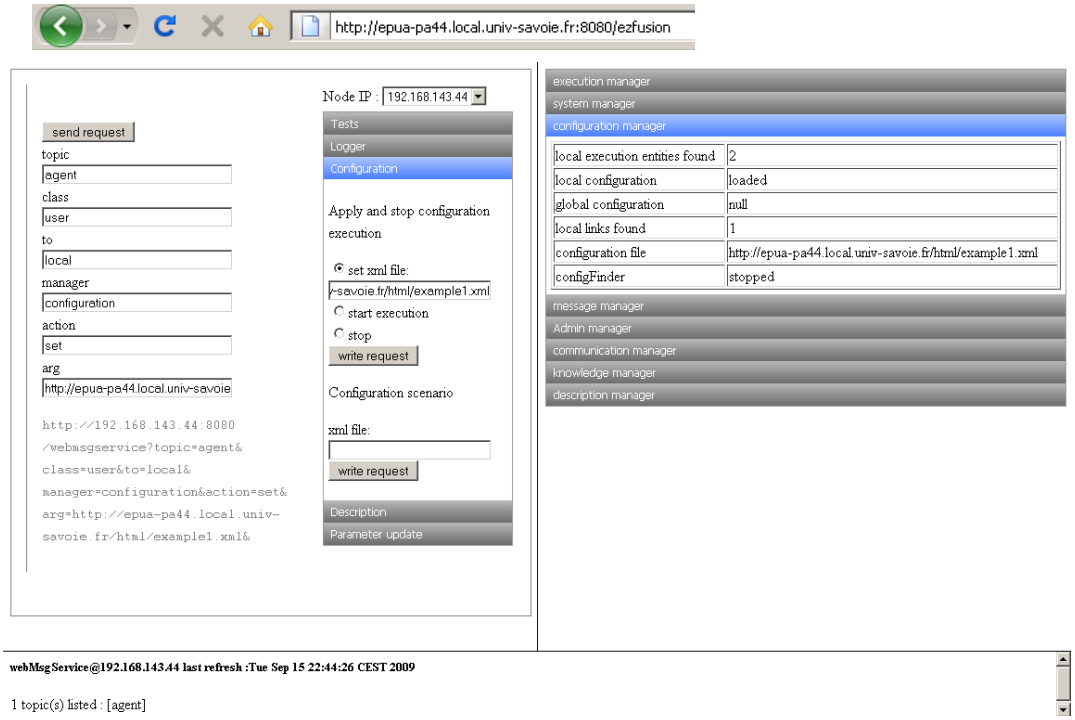


FIG. 13.3 – Interface Web de eZFusion.

- *Logger* : ce menu permet de spécifier le niveau des *log* affichés par le système à la page `http://127.0.0.1:8080/ezlog`,
- *Configuration* : ce menu actionne la lecture d'un fichier de configuration ainsi que le démarrage et l'arrêt du traitement,
- *Description* : ce menu permet la lecture d'un fichier de description du processus de fusion et sa diffusion dans le système,
- *Parameter update* : le concepteur du processus de fusion peut ajuster la valeur des paramètres des fonctions de fusion déployées dans le système.

Chaque menu permet de générer une requête (à gauche des menus) qui doit ensuite être envoyées au système de contrôle. Les résultats sont alors visibles dans l'interface d'information.

Interface d'information Cette interface regroupe toutes les informations relatives au cadre d'exécution qui héberge une instance d'eZFusion, toutes regroupées dans huit menus :

- *Execution manager* : ce menu renseigne sur les nœuds de fusion et les liens déployés sur le cadre d'exécution,
- *System manager* : ce menu présente les archives Java découvertes par le système de contrôle,
- *Configuration manager* : ce menu reprend le nom du fichier utilisé pour configurer le cadre d'exécution,
- *Message manager* : ce menu présente la partie du système de contrôle abonnée au service de messagerie interne à eZFusion,
- *Admin manager* : ce menu permettra à terme d'effectuer des opérations d'administration du cadre d'exécution,
- *Communication manager* : ce menu renseigne sur les interfaces réseau détectées par le système de contrôle,
- *Knowledge manager* : ce menu permet de visualiser l'utilisation des nœuds de fusion lors

de l'exécution du processus,
- *Description manager* : ce menu reprend le nom du fichier de description lu par le système.

13.3.2 Rédaction d'une configuration

Le démonstrateur permet actuellement le déploiement d'une configuration rédigée manuellement. Une configuration du système décrit quels sont les nœuds de fusion à déployer sur chaque cadre d'exécution. L'implémentation d'un lien entre deux ports de nœuds de fusion est explicitée, tout comme le choix du code exécutable utilisé pour appliquer le processus de fusion.

Chaque cadre d'exécution impliqué dans le déploiement du processus doit ainsi connaître la partie de la configuration qui le concerne. Pour cela, un fichier XML propre à chaque cadre d'exécution est rédigé puis transmis au système de contrôle *via* l'interface de commande et le menu *Configuration*.

Un fichier de configuration est composé de deux parties qui décrivent d'une part les fonctions de fusion à déployer sur le cadre, et d'autre part comment implémenter les liens entre les ports. La Figure 13.5 reprend l'exemple numéro 3 disponible sur les dépôts du projet. Cet exemple propose le déploiement du processus de fusion, illustré par la Figure 13.4, composé de trois sources d'information et d'un actionneur.

function Les éléments *function* du fichier XML décrivent les nœuds de fusion déployés grâce aux attributs suivants :
- *name* : le nom logique (unique) de la fonction de fusion,
- *classname* : le nom complet de la classe qui Java qui implémente la fonction de fusion,
- *jarfilename* : le chemin et le nom de l'archive Java qui contient la classe Java précédente
- *jardirname* : le chemin d'un dossier contenant plusieurs archives Java nécessaire à la classe Java précédente,
- *output name* : nom d'un port de résultat de la fonction de fusion,
- *input name* : nom d'un port de donnée de la fonction de fusion,
- *parameter name* : nom d'un port de paramètre de la fonction de fusion,

Les attributs *jarfilename* et *jardirname* ont une utilisation exclusive est le système renvoie un message d'alerte si les deux sont définis.

link Les éléments *link* du fichier XML décrivent l'implémentation d'un lien entre deux ports. Un lien peut ainsi être local si les deux nœuds de fusion connectés sont sur le même cadre d'exécution, ou distant dans le cas d'une affectation des nœuds de fusion sur deux cadres d'exécution.

Certains attributs des liens sont communs quelque soit leur type :
- *prodfname* : le nom de la fonction productrice,
- *prodpname* : le nom du port de résultat,
- *consfname* : le nom de la fonction consommatrice,
- *conspname* : le nom du port de donnée.

Un lien local est caractérisé par les attributs suivants :
- *local* : avec la valeur *true*,
- *remoteip* et *remoteport* : deux chaînes de caractères vides.

Un lien distant est composé de deux parties. La partie déployée sur le cadre où est affectée la fonction consommatrice est décrite par les attributs suivants :
- *local* : avec la valeur *false*,
- *remoteip* et *remoteport* : deux chaînes de caractères vides.

Alors que la partie déployée sur l'autre cadre d'exécution est décrite par les attributs suivants :
- *local* : avec la valeur *false*,

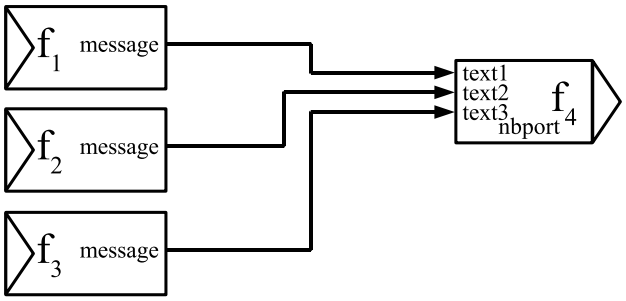


FIG. 13.4 – Exemple de processus de fusion : 3 sources d'information reliées à 1 actionneur.

- *remoteip* : l'adresse IP du cadre d'exécution où est affectée la fonction consommatrice,
- *remoteport* : le port IP du cadre présent à l'adresse *remoteip*.

13.3.3 Déploiement d'une configuration

Le déploiement d'une configuration nécessite l'acquisition de chaque fichier XML de configuration par les cadres d'exécution correspondants. Cette étape peut se dérouler selon deux modes opératoires, manuel ou par scénario, le mode scénario n'ayant d'intérêt que lors du déploiement d'une configuration sur plusieurs cadres d'exécution.

Déploiement manuel

Nous supposons que eZFusion est démarré sur le(s) cadre(s) d'exécution ; dans le cas contraire veuillez vous reporter au Paragraphe 13.3.1 pour lancer eZFusion sur un cadre d'exécution.

Sur chaque cadre d'exécution, se connecter à l'interface Web et ouvrir le menu *Configuration*. Ensuite, cocher le bouton *set XML file* puis entrer le chemin complet du fichier XML de configuration. Pour un fichier local sous Windows taper `file:///L:/mondossier/maconfig.xml` et pour un fichier local sous Linux taper `/mon/dossier/maconfig.xml`. Un fichier accessible par le réseau sera enregistré en tapant `http://mon.serveur.fr/mondossier/maconfig.xml`. Générer la requête correspondante en cliquant sur le bouton *write request*, puis cliquer sur le bouton *send request*.

L'interface d'information doit être mise à jour et le menu *Configuration manager* doit afficher le nom du fichier XML qui contient la configuration du cadre d'exécution.

Pour lancer le traitement, cocher le bouton *start execution* du menu *Configuration* (à gauche), générer puis envoyer la requête.

L'interface d'information doit être mise à jour et le menu *Execution manager* doit afficher le nom des fonctions de fusion déployées sur le cadre d'exécution.

Déploiement par scénario

Un scénario est un fichier XML qui contient une suite de commandes de contrôle. Ce fichier (Figure 13.6) est lu une fois par un des cadres d'exécution et les commandes sont envoyées par ce cadre aux autres cadres d'exécution.

message Chaque élément *message* correspond à une commande envoyée par le cadre d'exécution qui déroule le scénario. Un message est caractérisé par les attributs suivants :

- *to* : est l'adresse IP du cadre destinataire de la commande,

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
    <function name="f1"
    classname="org.ezfusion.tutorial.example1.TextProducer"
    jarfilename="file:///F:/ezfusion_tut.jar">
        <output name="message"></output>
    </function>
    <function name="f2"
    classname="org.ezfusion.tutorial.example1.TextProducer"
    jarfilename="file:///F:/ezfusion_tut.jar">
        <output name="message"></output>
    </function>
    <function name="f3"
    classname="org.ezfusion.tutorial.example1.TextProducer"
    jarfilename="file:///F:/ezfusion_tut.jar">
        <output name="message"></output>
    </function>
    <function name="f4"
    classname="org.ezfusion.tutorial.example3.MultiTextConsumer"
    jarfilename="file:///F:/ezfusion_tut.jar">
        <input name="text1"></input>
        <input name="text2"></input>
        <input name="text3"></input>
        <parameter name="nbport" class="java.lang.Integer" value="3"></parameter>
    </function>
    <link local="true" remoteip="" remoteport=""
    prodfname="f1" prodpname="message"
    consfname="f4" conspname="text1" >
    </link>
    <link local="true" remoteip="" remoteport=""
    prodfname="f2" prodpname="message"
    consfname="f4" conspname="text2" >
    </link>
    <link local="true" remoteip="" remoteport=""
    prodfname="f3" prodpname="message"
    consfname="f4" conspname="text3" >
    </link>
</configuration>
```

FIG. 13.5 – Exemple de fichier de configuration : ici trois sources d'information et un actionneur.

```
<scenario>
    <message to="192.168.143.51" topic="agent"
    class="user" manager="execution" action="stop"></message>
    <message to="192.168.143.50" topic="agent"
    class="user" manager="execution" action="stop"></message>
    <message to="192.168.143.50" topic="agent"
    class="user" manager="configuration" action="set"
    arg="http://mercure.local.univ-savoie.fr/sysrep/manipcam/maniphouse_cfg2_pc2.xml">
    </message>
    <message to="192.168.143.50" topic="agent"
    class="user" manager="execution" action="apply+config"></message>
    <message to="192.168.143.51" topic="agent"
    class="user" manager="configuration" action="set"
    arg="http://mercure.local.univ-savoie.fr/sysrep/manipcam/maniphouse_cfg2_pc1.xml">
    </message>
    <message to="192.168.143.51" topic="agent"
    class="user" manager="execution" action="apply+config"></message>
</scenario>
```

FIG. 13.6 – Exemple de fichier de scénario : ici le déploiement d'une configuration sur deux cadres d'exécution.

- *topic* : correspond au sujet de la messagerie interne à eZFusion lu par le cadre d'exécution du cadre destinataire,
- *class* : doit ici être égale à « user » car la commande est à l'initiative de l'utilisateur du système,
- *manager* : correspond à l'un des modules de gestion exécutés par l'agent qui contrôle le cadre destinataire,
- *action* : est la commande à exécuter,
- *arg* : est l'éventuel argument qui complète la commande à exécuter.

La liste complète des commandes possibles sera communiquée dans un rapport technique publié sur un des dépôt du projet.

13.3.4 Contrôle de l'exécution du processus

Les deux axes de contrôle du système sont partiellement accessible à l'utilisateur d'eZFusion avec d'une part la visualisation de l'utilisation des nœuds de fusion déployés sur un cadre d'exécution donné et d'autre part la modification de la valeur des paramètres des nœuds de fusion.

Le contrôle du système est donc actuellement réalisé par l'opérateur qui consulte un panneau de visualisation dans l'interface d'information. Ce panneau, illustré par la Figure 13.8, présente les temps de traitement des nœuds de fusion déployés sur le cadre d'exécution contrôlé, et présente également les taux d'utilisations de ces nœuds. Les taux d'utilisations sont basés sur les intervalles de temps de traitement et d'attente (le Chapitre 11 donne plus de détails).

L'utilisateur peut agire sur le second axe de contrôle, dédié au processus de fusion, par la mise à jour des paramètres des nœuds de fusion déployés sur le cadre d'exécution contrôlé. Le panneau de contrôle *Parameter update* de l'interface de contrôle, présenté sur la Figure 13.9, permet la saisie du nom du nœud de fusion et du nom du paramètre à mettre à jour avec la valeur entrée. Les valeurs des paramètres des nœuds de fusion déployés sur un cadre d'exécution peuvent être visualisées dans le panneau *execution manager* de l'interface d'information.

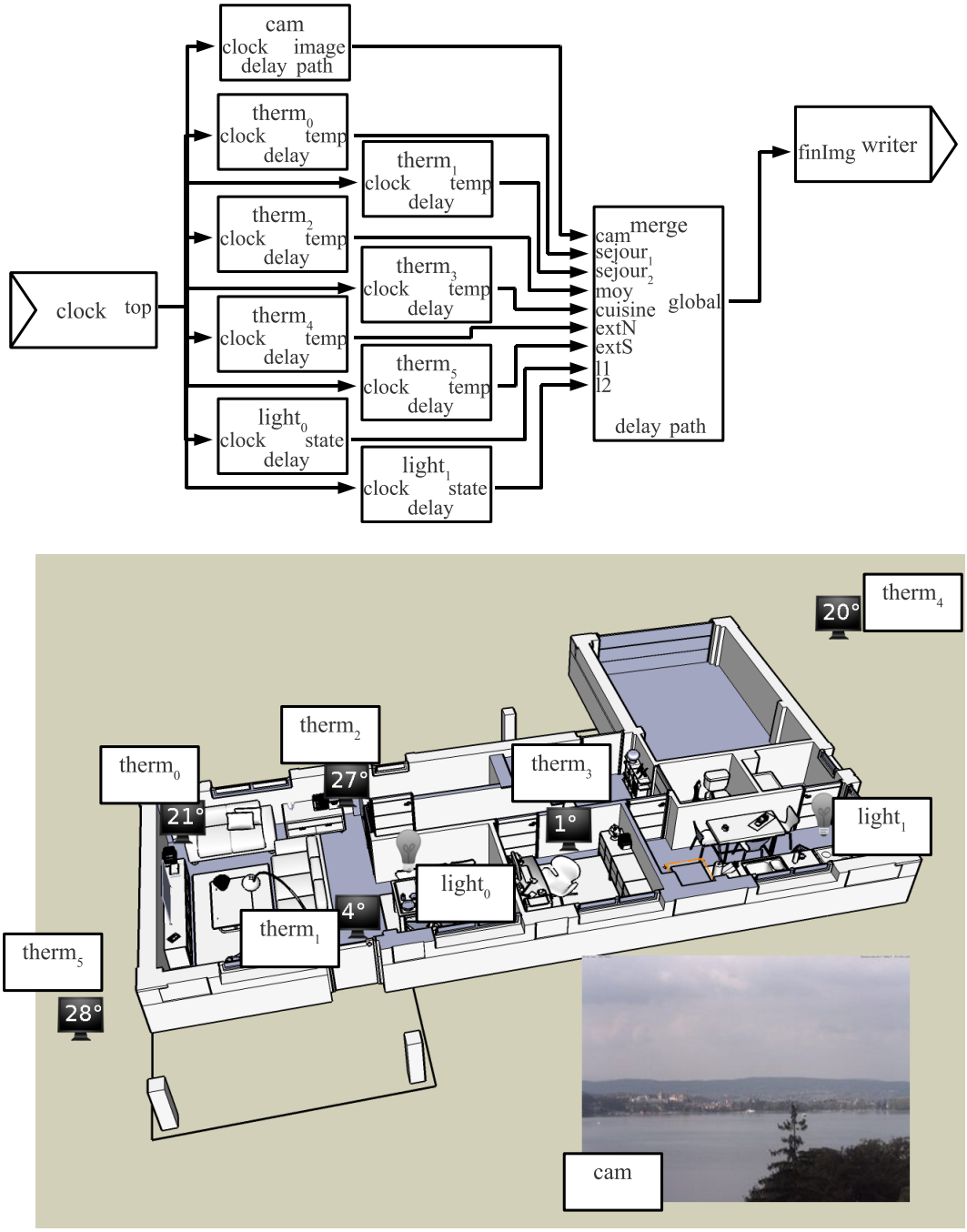


FIG. 13.7 – Exemple de processus de fusion : ici un exemple liée à la domotique, un haut le processus de fusion et en bas le résultat sur lequel nous avons repéré certains des éléments du processus.

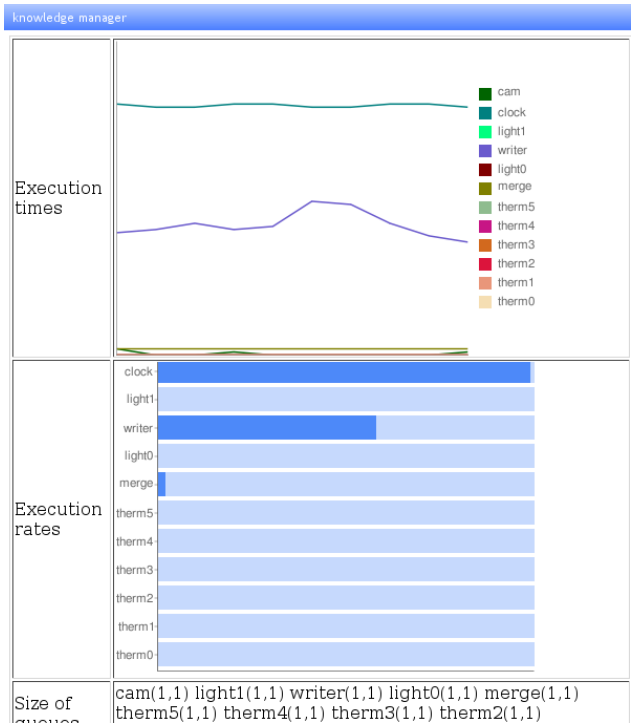


FIG. 13.8 – Interface d'information d'eZFusion : exemple de visualisation des temps de traitement (en msec) et des taux d'utilisation de nœuds de fusion.

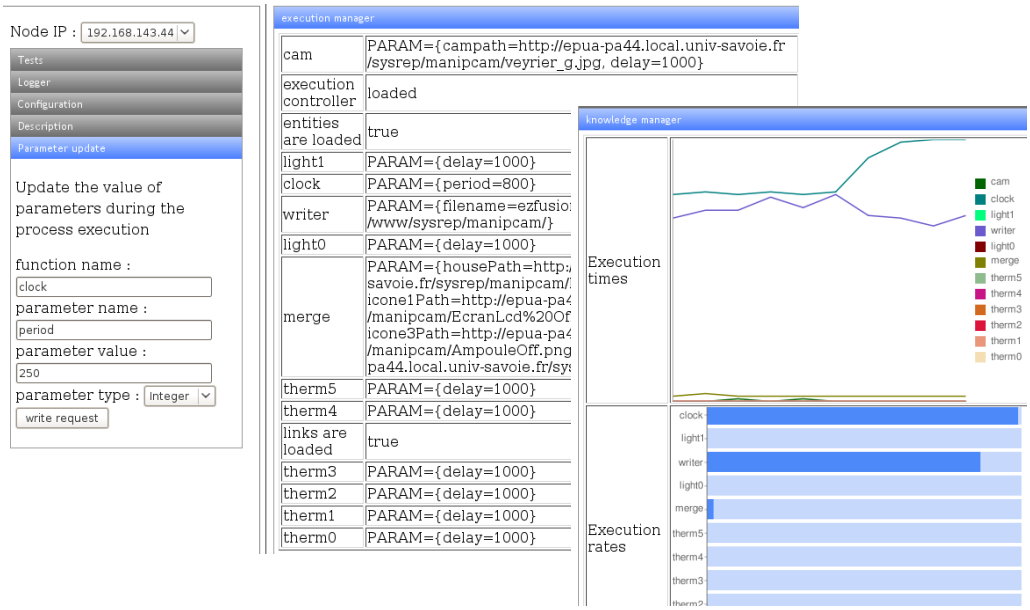


FIG. 13.9 – Interface d'information d'eZFusion : mise à jour de la valeur d'un paramètre du nœud de fusion *clock* et visualisation du changement au niveau du temps d'exécution.

Exemple L'exemple que nous proposons pour illustrer le contrôle du processus et du système est repris sur la Figure 13.7. Le processus de fusion déployé est le résultat d'un test de faisabilité dans le cadre d'un projet d'étude de la consommation énergétique d'habitations construites avec plusieurs matériaux témoins.

Le but ici est de montrer qu'il est possible d'appliquer le processus de fusion directement sur les mesures transmises par le site d'expérimentation, ou sur un historique de mesures, ou encore sur des mesures simulées. La connexion entre notre système et le site d'expérimentation n'a pas encore eu lieu et le processus de fusion que nous proposons n'a, pour l'heure, était testé que sur des données simulées.

Le processus de fusion que nous déployons dans cet exemple est composé de sept capteurs dont les mesures sont fusionnées au niveau de la fonction *merge*. Dans cet exemple, les capteurs recoivent en entrée un top horloge envoyé par la source de donnée *clock* afin de synchroniser la prise de mesure. Cette horologe pourrait être supprimée et une fréquence d'échantillonage serait alors établie pour chaque capteur au niveau de leurs paramètres. Le résultat de la fonction fusion *merge*, représenté dans la Figure 13.7, est finalement écrit dans le système de fichiers.

13.4 Administration

13.4.1 Gérer les plates-formes OSGi

L'administration d'eZFusion commence par la gestion des plates-formes où est installé le logiciel. Notre système dépend de deux services qui doivent être enregistrés sur chaque plate-forme :

- le service HTTP : ce service est déployé par le *bundle HTTP-Server* installé par défaut sur l'implémentation Knopflerfish d'OSGi,
- le service *RemoteOSGi* : le *bundle* qui déploie ce service n'est pas toujours installé par défaut et nous en proposons une version sur les dépôts d'eZFusion.

Concernant la configuration par défaut des services nécessaires à eZFusion, il convient de s'assurer qu'aucun d'eux n'entre en conflit avec un service déjà déployé, notamment ceux déployés hors de la plate-forme OSGi. En effet, la configuration par défaut du service HTTP déploie les *servlet* d'eZFusion sur le port 8080. De même, ROSGi est déployé sur le port 9278 et utilise des communications RMI qui peuvent interférer avec d'autres applications. L'administrateur du cadre d'exécution doit donc s'assurer de la compatibilité du système avant d'installer eZFusion.

13.4.2 Gérer la compatibilité des fonctions de fusion

Un des rôles de l'administrateur du système consiste à installer des bibliothèques nécessaires au déploiement d'un processus de fusion. Cette installation peut avoir lieu à deux niveaux. Ainsi, une archive Java qui contient des classes Java utilisées par un nœud de fusion du processus doit être placée à un emplacement accessible par le cadre d'exécution qui déploie ce nœud.

Ce fichier peut dans un premier cas être dans le système de fichier et il faudra alors préciser le chemin complet du fichier dans le fichier XML de configuration. Durant cette étape, il faudra prêter une attention particulière au format du chemin, propre au système de fichier.

La seconde option nécessite la copie de l'archive Java dans l'arborescence d'un serveur Web accessible par le réseau. Le chemin du fichier sera alors son URL commençant par `http://`.

Bibliothèques natives Enfin, certaines fonctions de fusion peuvent être implémentées par une classe java faisant appel à une bibliothèques native, comme un fichier *.dll* ou *.so*. Ces fichiers doivent alors être accessibles dans le système de fichier du cadre d'exécution.

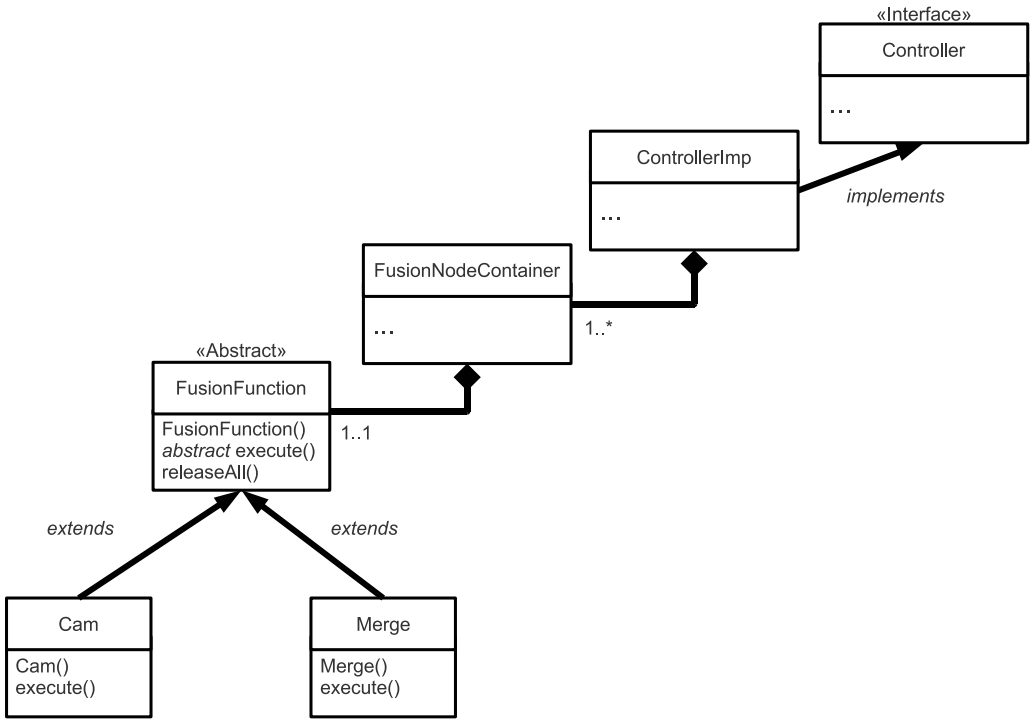


FIG. 13.10 – Diagramme de classes UML d'eZFusion : résumé de l'utilisation d'une implémentation de fonction de fusion, ici avec deux exemples.

13.4.3 Gérer les attributs du système

Ce dernier paragraphe dédié à l'administration d'eZFusion présente une fonctionnalité qui sera présente dans la prochaine version de notre système. En effet, l'administrateur devra prochainement mettre à jour deux attributs requis pas le système de contrôle.

Les plages d'adresses IP susceptibles d'héberger un cadre d'exécution sont en effet nécessaires au mécanisme de construction du graphe de connexion du réseau de cadres d'exécution.

Les autres attributs du système concernent :

- les emplacements d'archives Java, dans le système de fichier ou sur le réseau, pouvant être utilisées par eZFusion,
- l'étalonnage de la puissance d'un cadre d'exécution par l'exécution d'un nœud de fusion témoin.

13.5 Développement de fonction de fusion

Le développement d'une fonction de fusion, ou plus exactement de l'implémentation d'une fonction de fusion, s'effectue pour l'instant en Java. Des tests concluants ont montré que l'utilisation de bibliothèques natives était possible mais le détail de ces tests ne sont pour l'heure pas assez approfondis.

Le développeur d'une fonction de fusion doit dans un premier temps se procurer l'archive Java du *bundle* d'eZFusion. En effet, ce fichier est nécessaire pour importer et compiler le code Java de la fonction.

13.5.1 Développement d'une fonction de fusion

L'implémentation d'une fonction de fusion étend la classe abstraite *FusionFuction* reconnue par le système d'exécution d'eZFusion. La classe *FusionFuction* est composée d'un constructeur et de deux méthodes dont une abstraite.

La méthode abstraite *execute()* doit être implémentée dans le code du développeur. Cette méthode possède deux arguments avec des structures de données propres à eZFusion, respectivement pour les paramètres de la fonction de fusion et pour les données d'entrée. Le résultat de la méthode *execute()* est également d'un type propre à eZFusion.

La méthode *releaseAll()* est optionnelle. Le développeur peut l'implémenter dans le code de la fonction de fusion si des instructions sont à exécuter avant la destruction du nœud de fusion : par exemple la fermeture d'une liaison série.

Enfin, le constructeur de la fonction de fusion reçoit deux arguments dont un de type propre à eZFusion. Cet argument correspond aux paramètres lus dans le fichier de configuration du cadre d'exécution. Le second argument ne doit pas être modifié, il correspond à d'autres informations du nœud de fusion.

Les structures de données `FInfo` et `FParam` sont utilisés pour stocker les résultats partiels entre les nœuds de fusion et les paramètres. La lecture d'un paramètre doit être réalisée comme suit :

```
int foo = -1; // initialisation
try{
   // cas d'une donnée d'entrée
   foo = ((Integer)input.getInfoOn("nomDuPort")).intValue();
   // cas d'un paramètre
   // foo = ((Integer)param.getParamOn("nomDuPort")).intValue();
}
catch(Exception e){
   foo = BAR; // valeur par défaut en cas d'erreur
}
```

Nous supposons que le développeur connaît le type des données qu'il manipule et nous supposons également que les données échangées sont sérialisables. Cette dernière hypothèse est nécessaire car le service de communication entre cadres d'exécution restreint le type des données échangées.

Le résultat d'une fonction de fusion doit également être stocké dans la structure de donnée `FInfo`. La construction du résultat se déroule en deux étapes : la création de deux vecteurs et la construction de l'objet échangé.

En effet, deux vecteurs sont utilisés pour stocker le nom des ports de résultats et les valeurs à transmettre. Le constructeur de la structure `FInfo` prend alors ces deux vecteurs en arguments. Voici comment stocker le résultat d'une fonction de fusion :

```
Vecteur/*<String>*/ pNames = new Vector(); // nom des ports
Vector/*<Object>*/ outputs = new Vector();// valeurs transmises
FInfo result = null;

pNames.add("resultat1"); outputs.add(foo);
pNames.add("resultat2"); outputs.add(bar);

try {
   result = new FInfo(pNames, outputs);
}
catch (Exception e) {
  // une erreur peut être levée si les deux vecteurs ne sont pas cohérents
   e.printStackTrace();
}
return result;
```

13.5.2 Installation et utilisation d'une fonction de fusion

L'installation d'une fonction de fusion dans eZFusion ne peut se faire que par l'exportation du code Java nécessaire dans une archive Java. Une fois encapsulée dans un fichier *.jar*, le développeur, ou l'administrateur, doit copier ce fichier à un emplacement accessible aux cadres d'exécution utilisés pour déployer le processus de fusion.

Développement de eZFusion

eZFusion est un projet mis à la disposition de la communauté et nous comptons l'améliorer grâce aux retours d'utilisations de collègues. Cette amélioration du système s'effectuera également par la prochaine publication du code source d'eZFusion. Pour l'heure, aucun dépôt publique de code source n'a été ouvert et la mise à disposition de ce type d'outils, vraisemblablement par un serveur *subversion*, aura lieu courant 2010.

Mots-clé : Fusion d'informations, contrôle des systèmes répartis, analyse automatique de performances, répartition bio-inspirée par système multi-agents

Résumé : La fusion d'informations est une discipline bien plus ancienne que l'informatique moderne, et on en retrouve des applications dès lors qu'il est nécessaire de regrouper des informations potentiellement imprécises ou incertaines. Initialement réalisée de façon mentale, la fusion d'informations nécessite à présent une mise en œvre sur un support d'exécution informatisé, dénommé système de fusion, dont les évolutions technologiques ont guidée la répartition des ressources selon des contraintes géographiques, physiques et de sécurité.

Cette thèse présente une étude du contrôle des systèmes répartis de fusion d'informations. Elle propose notamment un système de contrôle capable d'adapter l'utilisation des ressources disponibles au processus de fusion (PF) mis en œvre, assez générique pour permettre l'expression des modèles de fusion les plus courants - en l'occurence des modèles de fusion basés sur des graphes de flot de données. Cinq tâches de contrôle sont dédiées à l'accomplissement d'un sous-objectif. La première consiste en la recherche d'une répartition des éléments du PF, basée sur le parcours d'un espace des répartitions possibles construit selon les ressources disponibles. Vient ensuite le déploiement effectif d'un PF sur un ensemble variable de ressources réparties. Pendant l'exécution du PF, la tâche d'analyse des performances du système permet d'adapter la répartition pour le rendre plus performant, grâce à un modèle de la répartition traduit en réseau de Petri stochastique généralisé. En parallèle, une autre tâche de contrôle surveille l'ensemble du système réparti pour détecter l'ajout de ressources, mais aussi les occurrences d'erreurs pouvant conduire à des défaillances. Enfin, la décision finale d'une modification de la répartition, soit pour en améliorer les performances, soit pour corriger la répartition courante et maintenir l'exécution du processus, découle d'une phase de négociation au sein d'un système multi-agents. L'ensemble du système de contrôle tire profit des ressources réparties par un mécanisme bio-inspiré, dans lequel la place de chaque agent est guidée par sa puissance de calcul et par ses interactions avec les autres agents du système. A titre expérimental, un système de fusion d'informations nommé eZFusion, implémentant les tâches de déploiement et d'analyse des performances, a été réalisé sur un réseau de plate-formes OSGI.

Abstract : Information fusion is almost as old as mankind and it is today a research domain of the computer science. Its use is required since potential uncertain or imprecise data should be combined. The fusion operations have to be performed on physical resources, named Information Fusion Systems (IFS), since fusion processes are now too complex. IFS are now distributed over communicating devices due to geographical or security issues. The purpose of this Ph. D. thesis deals with the control of distributed IFS. Especially it presents a control system able to adapt the use of available resources to the fusion process requirements. Thus, the description of the fusion process, built upon a data flow graph, is generic enough to cover the scope of the common information fusion models. Five control tasks compose the control system in order to reach five dedicated goals. The first task is in charge of matching each part of the process to the compatible resources. Then, the configuration built from the previous assignment, is effectively deployed on the distributed devices. During the execution of the fusion process, two other tasks operate : the first one analyses the system performance in order to improve it, and the second one monitors the execution and detects faults before failures occur. The final decision about updating the configuration comes from a negotiation between agents. The whole control system makes profit of the distributed resources thanks to a bio-inspired mechanism in which each agent infers its location in a control hierarchy from its CPU power and from its interactions with its neighbours. From a practical viewpoint, two tasks of the control system, respectively the one in charge of the deployment and the one in charge of the performance analysis, are implemented in eZFusion that is a distributed IFS developed over a network of OSGi frameworks.